

UNIVERZA V LJUBLJANI  
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Blaž Urankar

**Spletna aplikacija za osebnostno rast**

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE  
STOPNJE RAČUNALNIŠTVO IN INFORMATIKA

Ljubljana, 2016



UNIVERZA V LJUBLJANI  
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Blaž Urankar

**Spletna aplikacija za osebnostno rast**

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE  
STOPNJE RAČUNALNIŠTVO IN INFORMATIKA

MENTOR: doc. dr. Aleš Smrdel

Ljubljana, 2016



Rezultati diplomskega dela so intelektualna lastnina avtorja in Fakultete za računalništvo in informatiko Univerze v Ljubljani. Za objavlanje ali izkoriščanje rezultatov diplomskega dela je potrebno pisno soglasje avtorja, Fakultete za računalniško in informatiko Univerze v Ljubljani ter mentorja.



Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Sodobni čas daje vedno več poudarka na človekovo fizično plat, zanemarja pa njegovo psihično plat. Ker je človek psihofizično bitje, je za njegovo dobro počutje potrebno poskrbeti za obe plati. V diplomskem delu razvijte aplikacijo, ki bo uporabnikom pomagala krepiti njihovo samozavest ter jim pomagala pri njihovi osebni rasti. Pri razvoju aplikacije na strani odjemalca uporabite tehnologije za oblikovanje strani v kombinaciji s tehnologijami, ki omogočajo izdelavo odzivnih in bogatih spletnih aplikacij. Na strani strežnika pa pri razvoju uporabite tehnologijo Node.js. Pri uporabi bo razvita aplikacija potrebovala tudi podatke, ki naj jih pridobi iz baze podatkov. V ta namen uporabite neko tehnologijo, ki omogočajo upravljanje s podatkovnimi bazami. Za to, da bo aplikacija funkcionalna pa poskrbite tudi za ustrezne vsebine. Razvito aplikacijo pa ponudite v testiranje tudi uporabnikom in zabeležite ter opišite njihove odzive na aplikacijo.





## IZJAVA O AVTORSTVU DIPLOMSKEGA DELA

Spodaj podpisani Blaž Urankar sem avtor diplomskega dela z naslovom:

*Spletna aplikacija za osebnostno rast*

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Aleša Smrdela,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 4. marca 2016

Podpis avtorja:



*Zahvaljujem se mentorju, ker si je vedno vzel čas zame, mi hitro odgovarjal na vprašanja in mi odlično svetoval. Zahvaljujem se tudi mojim prijateljem, ki mi vsak dan znova pokažejo zabavno plat življenja. Brez njih ne bi prilezel tako daleč. Najbolj se moram zahvaliti svojim staršema. Ko v študiju nisem videl smisla, sta mi ga uspela pokazati. Ta diploma je za vaju.*







# Kazalo

**Povzetek**

**Abstract**

<b>Poglavje 1</b>	<b>Uvod .....</b>	<b>1</b>
<b>Poglavje 2</b>	<b>Uporabljena orodja in tehnologije .....</b>	<b>3</b>
2.1	Node.js, NPM in Express.js .....	4
2.2	MongoDB in Mongoose.js.....	4
2.3	Passport in JWT .....	5
2.4	AngularJS.....	5
2.5	JavaScript in jQuery.....	6
2.6	HTML in CSS .....	7
2.7	Git .....	7
<b>Poglavje 3</b>	<b>Zamisel.....</b>	<b>8</b>
3.1	Zasnova funkcionalnosti .....	8
<b>Poglavje 4</b>	<b>Podatkovna baza.....</b>	<b>11</b>
4.1	Entitete .....	11
4.2	Podatkovna shema .....	12
4.3	Seme podatkovne baze.....	13
<b>Poglavje 5</b>	<b>Strežniški del.....</b>	<b>15</b>
5.1	Strežniške poti.....	15
5.2	Glavni pogled.....	17
<b>Poglavje 6</b>	<b>Odjemalni del.....</b>	<b>19</b>
6.1	Storitve.....	19
6.2	Krmilniki.....	19
6.3	Direktive .....	20

6.4	Predloga rezina .....	21
6.5	Pogledi.....	22
6.5.1	Vstopni pogled .....	23
6.5.2	Pogleda za vpis in registracijo .....	23
6.5.3	Pogled profila .....	24
6.5.4	Pogled rezin.....	26
6.5.5	Pogled ustvarjanja rezin .....	28
6.5.6	Pogled profila sočutilca.....	29
6.6	Osnovne rezine iz aplikacije.....	30
<b>Poglavje 7</b>	<b>Sklepne ugotovitve .....</b>	<b>43</b>
7.1	Odzivi testnih uporabnikov .....	45
7.2	Cilji za nadaljnji razvoj .....	46



## Seznam uporabljenih kratic

kratica	angleško	slovensko
<b>JWT</b>	JavaScript Web Token	Spletni žeton JavaScript
<b>HTML</b>	HyperText Markup Language	Jezik za označevanje nadbesedila
<b>CSS</b>	Cascading Style Sheets	Kaskadne slogovne pole
<b>SQL</b>	Structured Query Language	Strukturiran jezik za poizvedovanje
<b>NoSQL</b>	Non-relational Structured Query Language	Strukturiran jezik za poizvedovanje brez relacij
<b>JSON</b>	JavaScript Object Notation	Zapis objektov JavaScript
<b>BSON</b>	Binary JavaScript Object Notation	Binarni zapis objektov JavaScript
<b>NPM</b>	Node Package Manager	Upravljaliec paketov Node
<b>MEAN</b>	MongoDB, Express.js, AngularJS, Nodejs	Kombinacija tehnologij MongoDB, Express.js, AngularJS, Nodejs



## Povzetek

**Naslov:** Spletna aplikacija za osebnostno rast.

V diplomski nalogi je predstavljen postopek zasnove in izdelave spletne aplikacije za osebnostno rast. Živimo v dobi, ko je za naše fizično udobje dobro poskrbljeno, v nevarnosti pa je naš um. Ljudje se pogosto počutijo manjvredne in ne razumejo svojih čustev, zato smo razvili spletno aplikacijo, ki pomaga krepiti samozavest in vzpodbuja razumevanje čustev. Razvijanja smo se lotili z uporabo nekaterih novejših tehnologij. Na strani strežnika smo uporabili tehnologije Node.js, Express.js, in MongoDB, na strani odjemalca pa AngularJS, HTML, CSS in JavaScript. Skozi leta smo nabrali veliko idej s psiholoških področij, ki v človeku vzbudijo pozitivna razkritja. Te ideje smo vključili v aplikacijo in jih predstavili v pričujočem delu. Predstavljajo osnovo za pozitivno okolje, ki raste skupaj s svojimi uporabniki. V diplomo smo dodali tudi sklepne ugotovitve, mnenja testnih uporabnikov in cilje za nadaljnje delo.

**Ključne besede:** spletna aplikacija, psihologija, osebnostna rast, čustva, samozavest.



## **Abstract**

**Title:** Web application for personal growth.

The thesis presents the process of designing and producing a web application for personal growth. We live in an era where it is well taken care of our physical comfort, but our mind is in danger. People often feel worthless and do not understand their feelings so we have developed a web application that helps strengthen confidence and encourages understanding of emotions. We started development with the use of some newer technologies. On the server side, we used Node.js, Express.js and MongoDB technologies while we used AngularJS, HTML, CSS and JavaScript on the client side. Over the years we have accumulated a lot of ideas from psychological fields that inspire positive revelations. We included these ideas in the application and we also presented them in this work. They provide the basis for a positive environment that grows together with its users. We also added conclusions, opinions of test users and objectives for future work into the thesis.

**Keywords:** web application, psychology, personal growth, feelings, self-esteem.



## Poglavje 1      Uvod

Živimo v dobi, ko je za naše fizično udobje dobro poskrbljeno. Morda celo preveč dobro. Glavni vzroki smrti so večinoma povezani z boleznimi, ki nastanejo zaradi pomanjkanja telesne aktivnosti, prekomerne teže in uživanja različnih substanc. Smrtonosnih plenilcev v razvitem svetu ni več, zato ni potrebe, da bi imeli telo pripravljeno za tek ali borbo za preživetje in redko kdo ga tudi ima, pa čeprav imamo na voljo več načinov kot kadarkoli prej, da pridobimo izklesano telo. Obiščemo lahko celo plastičnega kirurga, ki nam dele obraza spremeni natanko tako, kot si želimo. Dozdeva se nam, da nevarnosti za nas ni, a realnost je drugačna. Pozabljamo na nekaj veliko bolj pomembnega, naš um. Možgani so del našega telesa in delujejo kot računalnik, um pa je program, ki predstavlja to, kar smo. Če možgane prizadene bolezen in ne delujejo več normalno, to vpliva na delovanje našega uma, vendar ga ne spremeni. Program je še vedno napisan enako. To velja tudi za psihološke motnje, kot sta bipolarnost in anksioza. Njune simptome lahko zdravimo s tabletami, a um se zaradi tega ne bo spremenil, zato ga je potrebno zdraviti drugače. Potrebno je spremeniti program, ki ga izvajajo naši možgani.

Tu nastane problem, saj ljudje pogosto ne razumejo kaj čutijo, zakaj to čutijo in kako si lahko pomagajo, čeprav je to razloženo v literaturi[1]. Premik iz osebnih na materialne vrednote je povzročil, da se ljudje počutijo prazne, brez vrednosti in posledično brez prave samozavesti. Za ekonomijo je to blagoslov, saj se s ponujanjem materialnih vrednot odlično zasluži, tu pa verjetno malokdo gleda na to, kaj je pravzaprav zdravo za človeka.

Odločili smo se, da glede tega nekaj storimo. V mladosti so nas, tako kot ogromno ljudi, ovirala različna mišljenja, strahovi in dvomi, s katerimi pa se nismo želeli sprijazniti, zato smo začeli s potjo čustvenega in osebnostnega razvoja. Na podlagi združevanja računalniških tehnologij in psiholoških dognanj, nabranih v zadnjih letih, smo se odločili narediti aplikacijo, katere namen je človeku okrepiti samozavest, ga spoznati z njegovimi čustvi in mu omogočiti osebnostni razvoj. To storimo s tako imenovanimi rezinami, ki vsebujejo vse potrebno, da človeku razložijo neko idejo, ga pozneje nanjo spominjajo ter mu podajo napotke, kako jo uresničevati v praksi. Na spletu je prisotnih vse več ljudi vse več časa, kar nas je vzpodbudilo, da naredimo aplikacijo široko dostopno.

Z namenom, da ljudem približamo aplikacijo, smo se jo odločili razviti kot spletno aplikacijo. Pri izdelavi aplikacije smo na strežniku uporabili tehnologije Node.js, kot izvajalno okolje, ogrodje Express.js, kot vmesnik med odjemalcem in podatkovno bazo MongoDB ter module Mongoose.js, Faker, Passport.js in JWT. Na strani odjemalca smo uporabili tehnologije Git za nadzor verzij izvirne kode in še JavaScript, jQuery, CSS in HTML, ki smo ga nadgradili z uporabo ogrodja AngularJS, ki uporabniku ponuja izredno hitro odzivnost aplikacije in funkcionalnosti, ki drugače niso možne.

Ker je psihologija zelo širok koncept in ponuja različne alternative, poleg tega pa je človekova notranjost nekaj, kar se ne da »popredalčkati«, smo dodali možnost, da uporabniki sami dodajajo rezine, ki so za tem na voljo tudi drugim uporabnikom. S tem želimo ustvariti okolje za osebno rast, ki raste skupaj s svojimi uporabniki.



## Poglavje 2 Uporabljena orodja in tehnologije

Med študijem smo se naučili osnove jezikov HTML in CSS ter nekaj malega o funkcionalnosti skriptnega jezika JavaScript. Zanima nas razvoj spletnih aplikacij, zato smo se odločili, da poglobimo obstoječa znanja ter uporabimo še kaj bolj konkretnega.

Za delovanje aplikacije na strani strežnika smo preizkusili več tehnologij, a smo se na koncu odločili za izvajalno okolje Node.js v kombinaciji z ogrodjem Express.js. Ključna prednost je ta, da se uporablja izključno JavaScript, kar omogoča, da aplikacijo razvijamo z le enim programskim jezikom. To se nam je, kot začetnikom, zdelo izredno pomembno, saj smo osnove JavaScripta že poznali. Tako smo se lahko bolj osredotočili na ustvarjalni vidik razvoja in seveda na pisanje psihološke vsebine.

Imeli smo že kar nekaj izkušenj z uporabo tehnologije MySQL, ki smo jih pridobili med študijem, in pisanje stavkov SQL nam ni bilo tuje, a vendarle se tokrat nismo odločili za tak pristop. Uporabili smo podatkovno bazo MongoDB, ki uporablja mehanizem NoSQL, za kar obstajajo trije razlogi. Prvi je ta, da se na internetu pogosto omenja izraz MEAN, ki v osnovi predstavlja razvijanje z uporabo tehnologije MongoDB in še treh drugih, Express.js, AngularJS in Node.js, za katere smo se odločili že pred tem. To vsekakor ni bil najbolj pomemben razlog, saj MongoDB brez problema deluje tudi v kombinaciji z drugimi tehnologijami. Drugi razlog je ta, da je mehanizem NoSQL v zadnjih letih s prihodom novih tehnologij postal izredno popularen in se ga, kot razvijalcu nove generacije, vsekakor splača imeti v repertoarju. Najbolj pomemben razlog pa je način, kako NoSQL deluje. V dokumente podatkovne baze lahko brez težav shranimo kakršnekoli podatke, prav tako ni potrebno vnaprej definirati sheme podatkovne baze, vendar lahko kadarkoli dodamo poljubno zbirko s poljubnimi podatki. Aplikacija Čutilko je bil naš prvi samostojni obsežni projekt, zato smo pričakovali ogromno sprememb pri načrtovanju podatkovne baze, tu pa se je resnično pokazala vrednost podatkovne baze MongoDB. Bazo smo lahko gradili sproti, ko smo šele odkrivali nove elemente aplikacije.

Za povezavo med MongoDB in Express.js smo uporabili modul Mongoose.js, s katerim lahko definiramo sheme dokumentov v podatkovni bazi, na voljo pa ima tudi ogromno koristnih metod za ustvarjanje in upravljanje s podatki.

Za delovanje aplikacije na strani odjemalca smo izbrali tehnologijo AngularJS, ki nadgradi kodo HTMLja in omogoča izredno odzivnost aplikacije. Mnenja smo, da ima za začetnika zelo zahtevno krivuljo učenja, a so rezultati vredni truda. Želeli smo uporabiti tudi kakšno popularno ogrodje za oblikovanje strani, vendar bi bila s tem naša aplikacija preveč podobna ostalim, zato smo se malo bolj poglobili v tehnologijo CSS in aplikacijo oblikovali po svoje.

Za nadzor različic programske kode smo uporabili sistem Git, ki omogoča, da v katerem koli trenutku shranimo trenutno stanje kode in se po potrebi pozneje v to stanje vrnemo. S tem smo prihranili ogromno časa, ko smo del kode nezaželeno spremenili ali pa z njo nismo bili zadovoljni.

## 2.1 Node.js, NPM in Express.js

Node.js[2] je izvajalno okolje, ki ga je leta 2009 razvil Ryan Dahl. Zgrajeno je na odprtokodnem gonilniku Chrome V8[3]. Njegova glavna lastnost je zmožnost interpretacije in izvršitve kode skriptnega jezika JavaScript, poleg tega je izredno hiter in učinkovit ter deluje na vseh najbolj popularnih operacijskih sistemih. Odličen je za uporabo v aplikacijah, ki podatke izmenjujejo v realnem času. Zmožen je visoke prepustnosti ogromnega števila dvosmernih povezav, kjer lahko s komunikacijo začne tako odjemalec kot strežnik. Deluje asinhrono, kar pomeni, da ob zahtevi ne blokira delovanje programa, temveč se posveti drugim opravilom, ko pa je zahteva zaključena, izvrši njeno izhodno metodo[4]. Leta 2011 je bil razvit tudi upravljalac paketov Node (angl. NPM)[5], ki se namesti sočasno z Node.js. Omogoča namestitve katerekoli JavaScript knjižnice, ki jo najdemo v njegovi ogromni zbirki. Vsebuje praktično vse, kar bi potrebovali za začetek kateregakoli projekta. Express.js[6] je najbolj popularno minimalistično ogrodje za Node.js, namenjeno izjemno hitremu razvoju strežniškega dela spletnih aplikacij, z ogromno koristnimi metodami in odlično podporo na spletu. Tako kot Node.js je napisan izključno v programskem jeziku JavaScript in deluje na vseh najbolj popularnih operacijskih sistemih. Z njim lahko ustvarimo enostranske, večstranske in hibridne mobilne aplikacije, izvršten pa je tudi pri delovanju kot vmesnik med podatkovno bazo in odjemalnim delom aplikacije.

## 2.2 MongoDB in Mongoose.js

MongoDB[7] je podatkovna baza, ki uporablja mehanizem NoSQL. Od relacijskih podatkovnih baz se med drugim razlikuje po tem, da namesto tabel in vrstic uporablja zbirke in dokumente. Zbirke vsebujejo množice dokumentov, ki so sestavljeni iz kombinacij ključev in vrednosti, natanko tako, kot pri JSON formatu. Sheme podatkovne baze se lahko prosto

spreminjajo in jih ni potrebno definirati. Podatke lahko namreč vstavljamo čisto poljubno, brez predhodnega definiranja in brez omejitev. Baza podatke shranjuje v formatu BSON, ki je le binarni prikaz formata JSON. MongoDB je leta 2009 postal na voljo kot odprtokodni projekt, ki je skozi leta postal izredno popularen[8]. Mongoose.js[9] je modul, ki deluje kot vmesnik med podatkovno bazo in strežniškim delom aplikacije. Omogoča modeliranje podatkov, definiranje shem ter uporabo metod za upravljanje s podatki. Njegova uporaba ni nujna, lahko bi uporabljali osnoven gonilnik podatkovne baze MongoDB, vendar je razvijanje tako veliko lažje, saj ni potrebno vedeti, kaj se dogaja v ozadju.

## 2.3 Passport in JWT

Passport[10] je vmesni sistem za avtentikacijo uporabnika na strežniku, ki nevpisanim uporabnikom preprečuje izvajanje izbranih zahtev (npr. GET). Danes obstaja več načinov, kako se uporabnik lahko vpiše. Passport uporablja različne strategije, med katerimi se razvijalec lahko odloči, ali jih bo dodal v aplikacijo. Od tipične lokalne strategije, ki zahteva vpis z uporabniškim imenom in geslom, do vpisa z uporabniškimi računi drugih aplikacij, kot so najbolj popularna socialna omrežja. Zaradi novega načina avtentikacije je danes pogosta uporaba žetonov. JWT (JavaScript Web Token)[11] uporablja digitalno podpisane objekte JSON za prenos pomembnih informacij med dvema stranema. Vsebuje vse potrebne podatke o uporabniku in je izredno majhne velikosti, kar omogoča hiter prenos. Žeton oziroma objekt JSON je sestavljen iz glave, ki po navadi vsebuje tip žetona in algoritem kodiranja, tovor, ki vsebuje ključne podatke, na primer ime in ključ avtorja, ter podpis, ki je produkt kodiranja in podpisovanja podatkov iz glave in jedra, združeno s skrivnostjo, ki je po navadi okoljska spremenljivka.

## 2.4 AngularJS

AngularJS[12] je ogrodje za izdelavo aplikacije na strani odjemalca. Njegova glavna zmožnost je razširitev sintakse HTML, ki omogoča razvoj in uporabo funkcionalnosti, ki drugače niso na voljo. Na voljo ima kar nekaj komponent, ki omogočajo razvoj posebnih funkcionalnosti. Mi smo jih uporabili le nekaj:

- **Storitev** je komponenta, ki omogoča, da je del programske logike preko injiciranja odvisnosti dostopna vsem drugim komponentam aplikacije. To pomeni, da vsebuje metode in objekte, do katerih dostopa več delov aplikacije, vrednost objekta pa se lahko z uporabo storitve tudi v realnem času spreminja.

- **Krmilnik** je komponenta, ki vsebuje večino programske logike aplikacije na strani odjemalca. Do objektov in metod krmilnikov lahko dostopamo kjerkoli znotraj kode HTML, kjer to primerno definiramo. Izvorna koda 2.4.1 prikazuje primer uporabe krmilnika znotraj dokumenta HTML.

Izvorna koda 2.4.1: Primer uporabe krmilnika znotraj dokumenta HTML.

```
<div ng-controller="KrmilnikProfila">
  ...
</div>
```

S tem imamo v sklopu določenega elementa HTML na voljo vse objekte in metode, ki so vezane na `$scope` krmilnika. Zaželeno je, da je programske logike znotraj elementov HTML čim manj.

- **Direktiva** je komponenta, ki predstavlja nek nov element HTML. Poleg tipičnih elementov, kot je `<div></div>`, lahko na primer ustvarimo direktivo `zanka`, ki nek stavek izpiše tolikokrat, kolikor je vrednost podane spremenljivke, uporabimo pa jo kot element `<zanka stevilo=10></zanka>`. Direktivo uporabimo za združevanje kode, ki se v aplikaciji pogosto pojavlja. V njej lahko povežemo izbran krmilnik z izbrano predlogo, ki vsebuje kodo HTML, ki ima potem možnost uporabe celotne funkcionalnosti krmilnika.
- **Pogled** je komponenta, ki vsebuje kodo HTML, ki v glavnem pogledu, `indeks.html` datoteki, nadomesti vrstico `<div ui-view></div>`. Deluje v smislu spletne strani aplikacije, s to razliko, da je pogled vedno le del glavne spletne strani, ki se v našem primeru nikoli ne spremeni. To omogoča izredno hitro menjavo vsebine, ki jo vidi uporabnik aplikacije, saj se glavna spletna stran nikoli ponovno ne naloži, temveč se le zamenja pogled.

AngularJS uporablja princip dvojne izmenjave podatkov, kar omogoča, da se spremenljivka krmilnika v realnem času spreminja tako v kodi HTML kot v samem krmilniku. Spremenljivke so v kodi HTML prikazane na način `{{spremenljivka}}`, kar v brskalniku prikaže njeno vrednost.

## 2.5 JavaScript in jQuery

JavaScript[13] je objektno usmerjen skriptni jezik in je med najbolj popularnimi programskimi jeziki na svetu. Z vzponom izvajalnega okolja Node.js so nastale tehnologije,

ki omogočajo uporabo JavaScripta na vseh področjih razvoja spletnih aplikacij, tako na strežniku kot pri odjemalcu, zato ni napačno reči, da je JavaScript programski jezik prihodnosti. Glavna zmožnost JavaScripta je implementacija interaktivnosti v aplikacijo, prav tako pa interpreterja zanj ni potrebno posebej nameščati, saj je vključen v večino najbolj popularnih brskalnikov. Ponuja ogromno že razvitih knjižnic, ki omogočajo uporabo funkcionalnosti, katerih delovanje v ozadju nam ni potrebno poznati. jQuery[14] je ena najbolj popularnih knjižnic JavaScript. Glavne prednosti njene uporabe so manjša količina kode, lahkotno upravljanje z elementi HTML, upravljanje dogodkov, animacije in podpora v vseh najbolj popularnih brskalnikih.

## 2.6 HTML in CSS

HTML[15] je označevalni jezik, ki definira strukturo spletnih dokumentov. To stori z elementi, ki zagotovijo postavitve teksta in slik v brskalniku. Elementi so sestavljeni iz značk, atributov in vsebine: `<značka atribut=vrednost>Vsebina<značka>`. Med tem, ko HTML skrbi za strukturo spletnih dokumentov, CSS[16] skrbi za obliko. To je stilski jezik, ki definira postavitve in izgled elementov HTML. Določa lahko na primer velikost, barvo, ozadje, izgled robov, odmike od drugih elementov, pisave in še marsikaj.

## 2.7 Git

Git[17] je sistem za upravljanje verzij izvirne kode. Omogoča, da ekipa razvijalcev deluje na istem projektu ob istem času. En član ekipe ima svojo lokalno verzijo projekta, ki jo je prenesel iz strežnika, kjer je naložena najnovejša verzija projekta. Ko član spremeni kodo, lahko novo različico pošlje nazaj na strežnik, od koder si jo prenesejo preostali člani ekipe, ob tem pa so obveščeni o spremembah, tako da lahko prilagodijo svojo različico. Git lahko uporablja tudi posameznik brez interneta na lokalnem računalniku ter si tako prepreči preglavice, ki se zgodijo ob ponesrečeni spremembi kode ali izgubi podatkov.

## **Poglavje 3      Zamisel**

Ideja spletne aplikacije Čutilko je uporabniku ponuditi psihično podporo, kadarkoli in kjerkoli jo potrebuje, če le ima dostop do interneta. Želimo si, da bi si lahko sčasoma uporabnik, ki ga imenujemo čutilec, podporo nudil sam, s pomočjo idej, ki se jih je na naši spletni strani naučil. S tem si bo okrepil samozavest, v trenutku, ko ga preplavijo čustva, pa bo razumel, zakaj se to dogaja in kako se s tem spoprijeti.

V pričakovanju osebnostne rasti mu želimo ponuditi tudi možnost pisanja lastnih idej, ki jih lahko deli z drugimi čutilci, ter tako postane ključni del pri nadaljnjem razvoju vsebine spletne aplikacije. Naš končni cilj je ustvariti neverjetno priložnost osebnostne rasti, h kateri bo s svojimi idejami prispeval sleherni čutilec oziroma uporabnik aplikacije.

Pri idejah je ključno, da jo človek dojame, ne pa da se jo nauči na pamet. Spletna stran mora biti na pogled prijetna in pomirjajoča, da čutilca ne motijo nepotrebni elementi ter je tako sproščen in bolj dovzeten za ideje.

### **3.1      Zasnova funkcionalnosti**

Čutilec po registraciji oziroma vpisu prispe na svojo profilno stran. To je njegov dom, kjer so izpisani njegovi dosežki, misel dneva, dodane ideje, ki jih imenujemo rezine, in nedavne aktivnosti sočutilcev, ki jih spremlja. Sočutilec je nek drug uporabnik aplikacije, ki ga lahko čutilec spremlja tako, da gre na njegovo profilno stran in klikne na gumb »Spremljaj«. Od takrat naprej se na profilni strani čutilca izpiše aktivnost vsakič, ko sočutilec ustvari novo rezino.

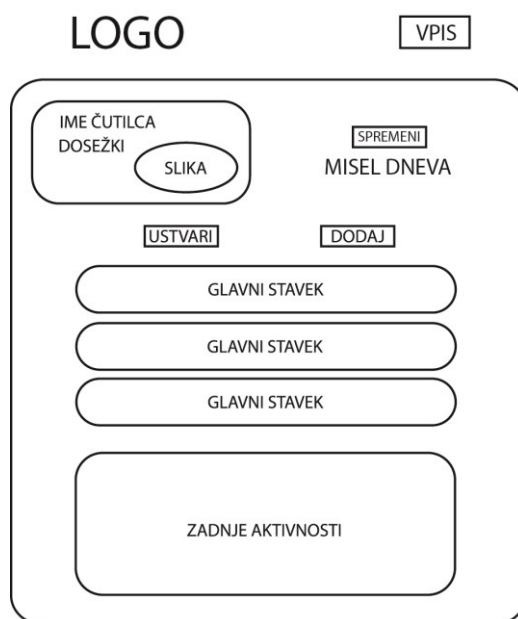
Dosežki so število, ki pove, koliko je čutilec napredoval v smislu osebnostne rasti. Vsakič, ko v svojo zbirko doda rezino, se le-tej poveča moč za eno točko, njemu in avtorju rezine pa se za eno točko povečajo dosežki. Čutilec je tako motiviran, da poleg dodajanja oziroma branja drugih rezin ustvari tudi svoje. S tem, ko si nabira dosežke, prejema različne profilne slike, ki pričajo o njegovem napredovanju.

Misel dneva si čutilcec napiše sam in je vidna vsakemu, ki njegov profil obišče. Odraža trenutno stanje čutilca, ki ga želi deliti z drugimi. Pričakujemo, da bodo misli iz dneva v dan bolj pozitivne.

Profil intervalno prikazuje po največ tri dodane rezine naenkrat. Ob vpisu oziroma registraciji v sistem profil vsebuje le eno rezino, ki razloži strukturo rezine in delovanje spletne strani. Ta rezina izgina, ko čutilcec doda novo, svojo prvo rezino.

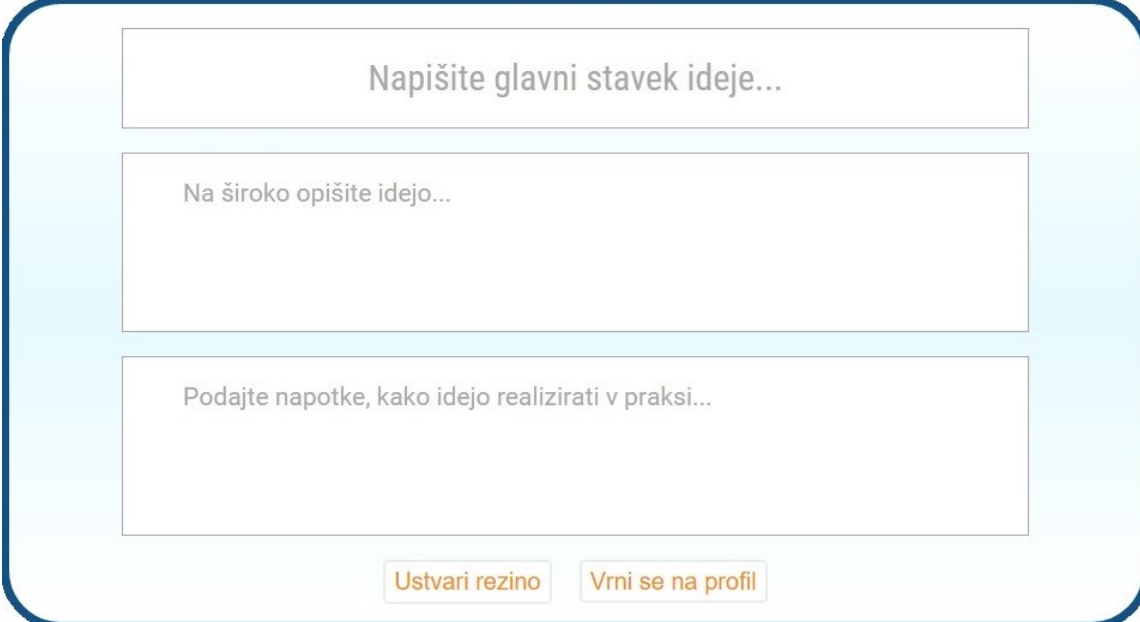
Rezine so sestavljene iz glavnega stavka, jedra in prakse. Glavni stavek je besedilo, ki čutilca spomni na idejo, ki jo rezina opisuje. Biti mora kratek, a dovolj smiselno napisan, da opravi svojo nalogo. V jedru je ideja razložena dovolj podrobno, da v bralcu vzbudi novo razkritje. To je najpomembnejši del rezine, saj bi brez njega imeli le stavek, ki bi morda res nekaj vzbudil v čutilcu, a ta ne bi razumel, zakaj je temu tako. Praksa poda praktične napotke, kako idejo v življenju uresničevati. Brez tega bi čutilcec idejo vseeno razumel, a ne bi imela enake vrednosti, saj je morda ne bi znal uporabljati.

Gumb »Dodaj« čutilca popelje na stran za dodajanje, kjer so na voljo vse rezine, ki jih čutilcec še nima v svoji zbirki dodanih rezin. Urejene so po moči rezine, ki pove, koliko čutilcev si jo je dodalo v svojo zbirko. Sprva je prikazan le glavni stavek, s klikom nanj pa se rezina odpre in izpiše podrobnosti. Pokažejo se jedro, praksa, ime avtorja, moč rezine in gumb za dodajanje. Ob kliku na ime avtorja rezine se odpre njegova profilna stran, kjer je vidna njegova misel dneva ter vse rezine, ki jih je ustvaril. Slika 3.1.1 prikazuje shematsko zasnovo profilne strani čutilca.



Slika 3.1.1: Shematska zasnova profilne strani čutilca.

Obstaja še stran za ustvarjanje rezin, prikazana na sliki 3.1.2, ki vsebuje obrazec, kjer čutilec vpiše glavni stavek, jedro in prakso. Ko rezino ustvari, je ta na voljo vsem sočutilcem, avtomatsko pa se mu doda tudi v lastno zbirko dodanih rezin.



Napišite glavni stavek ideje...

Na široko opišite idejo...

Podajte napotke, kako idejo realizirati v praksi...

Ustvari rezino Vrni se na profil

Slika 3.1.2: Obrazec za ustvarjanje rezin.



## Poglavje 4 Podatkovna baza

Preden smo se lotili načrtovanja podatkovne baze smo imeli izkušnje le s sistemom MySQL, ki uporablja medsebojno povezane entitete. Entitetni tip je koncept neke celote z določenimi lastnostmi, ki obstaja v shemi podatkovne baze in je v razmerju z nekim drugim entitetnim tipom, entiteta pa je posamezen primer tega entitetnega tipa. Primer sta avto in človek. Avto ima na primer svojo barvo, končno hitrost in ceno, lastnik pa ime in priimek. Ko konkreten človek kupi konkreten avto, postane lastnik tega avta in entiteti sta v razmerju.

### 4.1 Entitete

Ker smo se zaradi večje možnosti prilagajanja odločili za sistem MongoDB, je bilo načrtovanje nekoliko drugačno. Posamezni primeri entitetnega tipa tokrat niso imenovani entitete, ampak dokumenti. Shema je bilo potrebno razviti na drugačen način, a vseeno smo določili entitetne tipe, kot bi to storili v MySQL:

- Edinstvene lastnosti entitetnega tipa **rezina** so čas nastanka (*cas*), stavek, jedro, praksa in moč (*moc*). Vsebuje tudi ključ avtorja (*idCutilca*) in njegovo ime (*imeCutilca*). Ključ avtorja ni neposredno povezan s katerim od drugih dokumentov, lahko pa z njim dokument ročno poiščemo.
- Glavni entitetni tip je **čutilec**. Njegove edinstvene lastnosti so ime čutilca (*imeCutilca*), vrednosti za kodiranje gesla (*hash in sol*), dosežki (*dosezki*) in *misel*, vsebuje pa tudi lastnosti, ki so pravzaprav drugi entitetni tipi ali pa njihove zbirke. Entitetni tip čutilec tako vsebuje še lastnosti *ustvarjeno in dodano*, ki sta zbirki dokumentov entitetnega tipa *rezina*, lastnost *zgodovina*, ki je zbirka dokumentov entitetnega tipa *aktivnost*, in še dve lastnosti *idji\_dodanih* ter *socutilci*, ki sta zbirki ključev dodanih *rezin* in *sočutilcev*, ki jih čutilec spremlja.
- Edinstvene lastnosti entitetnega tipa **aktivnost** so čas nastanka (*cas*) in dejanje, vsebuje pa še lastnosti drugih entitetnih tipov *idCutilca*, *imeCutilca* in objekt entitetnega tipa *rezina*.

Tak pristop smo uporabili, ker MongoDB podatke shranjuje v formatu BSON, ki je binarno predstavljen format JSON. To nam omogoča, da posamezen primer entitetnega tipa shranimo kot dokument z določenimi lastnostmi, te lastnosti pa so lahko tudi posamezni primeri oziroma objekti drugih entitetnih tipov.

## 4.2 Podatkovna shema

V aplikaciji se vse odvija okrog čutilca. Brez njega ne bi bilo vsebine, zato ima aplikacija le en podatkovni model, *Cutilec*. Znotraj je definirana shema *CutilecSchema*, ki je instanca objekta *mongoose.Schema()* iz modula *Mongoose.js*. Izvorna koda 4.2.1 prikazuje shemo podatkovnega modela *Cutilec*, ki upošteva vse entitetne tipe.

Izvorna koda 4.2.1: Shema podatkovnega modela *Cutilec*.

```
var CutilecSchema = new mongoose.Schema({
  _id: {type: Number, unique: true},
  imeCutilca: {type: String, unique: true},
  hash: String,
  sol: String,
  zgodovina: [{
    cas: { type : Date, default: Date.now },
    idCutilca: Number,
    imeCutilca: String,
    dejanje: String,
    rezina: {
      _id: String,
      idCutilca: Number,
      imeCutilca: String,
      stavek: String,
      jedro: String,
      praksa: String,
      moc: {type: Number, default: 1}
    }
  ]},
  dosezki: {type: Number, default: 1},
  ustvarjeno: [{
    _id: String,
    cas: { type : Date, default: Date.now },
    idCutilca: Number,
    imeCutilca: String,
    stavek: String,
    jedro: String,
    praksa: String,
    moc: {type: Number, default: 1}
  ]},
  dodano: [{
    _id: String,
    idCutilca: Number,
```

```
    imeCutilca: String,  
    stavek: String,  
    jedro: String,  
    praksa: String,  
    moc: {type: Number, default: 1}  
  }],  
  idji_dodanih: [String],  
  misel: {type: String, default: 'Kliknite tu in napiši  
te misel, ki se ujema z vašim trenutnim počutjem.'},  
  socutilci: [  
    {  
      _id: Number,  
      imeCutilca: String  
    }  
  ]  
});
```

Lastnost `_id` oziroma ključ je avtomatsko dodan vsakemu dokumentu, razen če je posebej definiran. Privzeto je tipa `ObjectID`, a tudi to lahko spremenimo. V našem primeru je ključ čutilca število, ključ rezine pa niz. Temu je tako, ker za ključ rezine uporabljamo kombinacijo ključa avtorja (čutilca), podčrtaja in zaporedno številko rezine v avtorjevi zbirki ustvarjenih rezin. Nastavljene so tudi nekatere privzete vrednosti lastnosti, na primer stavek za začetno misel in moč rezine ob njenem nastanku. Poleg sheme so definirane tudi njene metode:

- Metoda `nastaviGeslo(geslo)` ustvari naključno vrednost za lastnost `sol`, ki se uporabi pri kodiranju gesla v vrednost lastnosti `hash`.
- Metoda `pravilnoGeslo(geslo)` uporabi že obstoječo vrednost lastnosti `sol` ter podano geslo spet zakodira. Novo dobljeno vrednost `hash` nato primerja z že obstoječo vrednostjo `hash` in vrne `true` ali `false`.
- Metoda `generirajJWT()` ustvari in vrne nov podpisan žeton za avtentikacijo.

Na koncu se z ukazom `mongoose.model('Cutilec', CutilecShema)` model dejansko ustvari in je na voljo v drugih delih aplikacije.

### 4.3 Seme podatkovne baze

`seme.js` je datoteka, ki prva uporabi model `Cutilec` in se požene le takrat, ko aplikacijo prvič poženemo na strežniku. Ustvari osnovnega čutilca, ki vsebuje vse rezine, ki smo jih ustvarili mi. Podatkovne baze sicer ni potrebno vedno napolniti z začetnimi podatki, a v tem primeru nam to prihrani čas, saj bi morali drugače osnovne rezine vedno znova ročno dodajati.

Preden smo napisali rezine, smo za testno vsebino uporabljali modul Faker, ki omogoča, da namesto pravih vrednosti uporabljamo naključno generirane vrednosti, ki so lahko različnih tipov. Izvorna koda 4.3.1 prikazuje način, kako lahko podatkovno bazo polnimo s testnimi podatki.

Izvorna koda 4.3.1: Sejanje podatkovne baze s testnimi podatki.

```
var Cutilec = mongoose.model('Cutilec');
var faker = require('faker');

var cutilec = new Cutilec();
cutilec._id = 1;
cutilec.imeCutilca = 'Blaž';
cutilec.nastaviGeslo('geslo');
cutilec.ustvarjeno = [{
  _id: '1_1',
  idCutilca: id,
  imeCutilca: cutilec.imeCutilca,
  stavek: faker.Lorem.sentence(),
  jedro: faker.Lorem.paragraph(),
  praksa: faker.Lorem.paragraph()
}];
```

## Poglavje 5     Strežniški del

Ob zagonu strežnika se vzpostavi povezava s podatkovno bazo MongoDB, ki avtomatsko ustvari bazo `cutilko`, če ta še ne obstaja, po želji pa lahko izvršimo tudi datoteko `seme.js`, ki podatkovno bazo sprazni in napolni s svežimi podatki. Uvozijo se privzeti moduli, za tem pa se nastavi lokacija za glavni pogled aplikacije in datoteka, ki vsebuje strežniške poti. Prav tako se uvozi modul `dotenv`, ki nam omogoča, da nastavimo okoljsko spremenljivko `SKRIVNOST`, ki se pozneje uporabi pri avtentikaciji.

### 5.1     Strežniške poti

V datoteki, ki vsebuje strežniške poti, se najprej uvozijo moduli `Mongoose.js`, za upravljanje s podatkovno bazo, `Passport`, za upravljanje z avtentikacijo, `JWT` za upravljanje z žetoni in `Express.js Router`, ki omogoča nastavljanje strežniških poti. Poleg tega se uvozi tudi že prej ustvarjeni model `Cutilec`.

Privzeta strežniška pot, prikazana v izvorni kodi 5.1.1, je najbolj enostavna in le poveže osnovni naslov spletne aplikacije z glavnim pogledom `indeks.html`. Definirane ima tip zahteve, spletni naslov in izhodno metodo, ki se izvrši ob uporabi strežniške poti.

Izvorna koda 5.1.1: Privzeta strežniška pot.

```
router.get('/', function(req, res, next) {  
  return res.render('indeks');  
});
```

Parameter `req` vsebuje podatke, ki jih pošlje odjemalec, `res` pa podatke, ki jih z uporabo ukaza `return` odjemalcu pošlje strežnik nazaj. Naloga parametra `next` je predajanje podatkov naslednjim morebitnih metodam, ki bodo podatke obdelovale. Tako izgleda vsaka strežniška pot, le da imajo lahko drugačno zahtevo in spletni naslov, v sami metodi pa se izvrši še več programske logike, preden metoda vrne neko vrednost.

V datoteki je poleg strežniških poti definirana še metoda `je_id_v_tabeli(id, tabela)`, ki preveri, ali je podan ključ v podani tabeli, in pa strežniška pot s posebno

zahtevo PARAM, ki vsakič, ko spletni naslov vključuje ključ čutilca, vnaprej pridobi njegove podatke, kar prikazuje izvorna koda 5.1.2.

Izvorna koda 5.1.2: Zahteva PARAM, ki vrne podatke o čutilcu.

```
router.param('cutilec', function(req, res, next, id) {

  Cutilec.findOne({
    _id: id
  }, function(err, cutilec) {
    if (err) {
      return next(err);
    }
    if (!cutilec) {
      return next(new Error('Na najdem čutilca.'));
    }
    req.cutilec = cutilec;
    return next();
  });
});
```

Strežnik vsebuje še naslednje strežniške poti:

- Pot z zahtevo GET na naslov `/api/v1/cutilci/:cutilec`, ki uporabi parameter `cutilec` iz naslova, da iz podatkovne baze potegne pravega čutilca ter ga vrne odjemalcu.
- Pot z zahtevo POST na naslov `/api/v1/rezine`, ki preko `req` dobi ključ čutilca in rezino. To rezino doda v čutilčevo tabelo dodanih rezin v podatkovni bazi in mu poveča dosežke za eno točko. Prav tako za eno točko poveča moč rezine in dosežke avtorja rezine. Odjemalcu vrne potrditev o uspehu.
- Pot z zahtevo GET na naslov `/api/v1/rezine/:cutilec`, ki uporabi parameter `cutilec` iz naslova, da iz podatkovne baze potegne pravega čutilca in odjemalcu vrne vse rezine, ki jih ta čutilec še nima v svoji zbirki dodanih rezin.
- Pot z zahtevo POST na naslov `/api/v1/rezine/:cutilec`, ki uporabi parameter `cutilec` iz naslova, da iz podatkovne baze potegne pravega čutilca. Preko `req` dobi rezino, ki jo je ta čutilec ustvaril, in jo doda v njegovo tabelo ustvarjenih rezin v podatkovni bazi. Prav tako jo doda v njegovo zbirko dodanih rezin, mu poveča dosežke za eno točko in v tabelo zgodovina doda novo aktivnost. Ta aktivnost vsebuje informacije o tem, kdaj je uporabnik ustvaril rezino in na katero rezino se to navezuje. Odjemalcu vrne potrditev o uspehu.

- Pot z zahtevo PUT na naslov `/api/v1/misel/:cutilec`, ki uporabi parameter `cutilec` iz naslova, da iz podatkovne baze potegne pravega čutilca. Preko `req` dobi novo misel, s katero v podatkovni bazi prepiše staro misel. Odjemalcu vrne potrditev o uspehu.
- Pot z zahtevo GET na naslov `/api/v1/zgodovina/:cutilec`, ki uporabi parameter `cutilec` iz naslova, da iz podatkovne baze potegne pravega čutilca in odjemalcu vrne zgodovino vseh drugih uporabnikov oziroma sočutilcev, ki jih ima v tabeli spremlja.
- Pot z zahtevo POST na naslov `/api/v1/socutilci/:cutilec`, ki uporabi parameter `cutilec` iz naslova, da iz podatkovne baze potegne pravega čutilca. Preko `req` dobi sočutilca, ki ga doda v čutilčevo tabelo dodanih sočutilcev. Odjemalcu vrne potrditev o uspehu.
- Pot z zahtevo POST na naslov `/api/v1/registracija`, ki preko `req` dobi ime in geslo čutilca. V kolikor eden od njiju manjka, vrne napako. Za tem preveri, koliko čutilcev je že v podatkovni bazi in novemu čutilcu dodeli ključ, ki je za ena večji od ključa zadnjega čutilca v podatkovni bazi, nato pa ga shrani v podatkovno bazo. Odjemalcu vrne žeton, ki ga generira s pomočjo metode `generirajJWT()` iz modela `Cutilec`.
- Pot z zahtevo POST na naslov `/api/v1/vpis`, ki preko `req` dobi ime in geslo čutilca. V kolikor eden od njiju manjka, vrne napako. Za tem uporabi metodo `authenticate()` iz modula `Passport`, ki preveri ustreznost podatkov, in odjemalcu vrne žeton, ki ga generira s pomočjo metode `generirajJWT()` iz modela `Cutilec`.

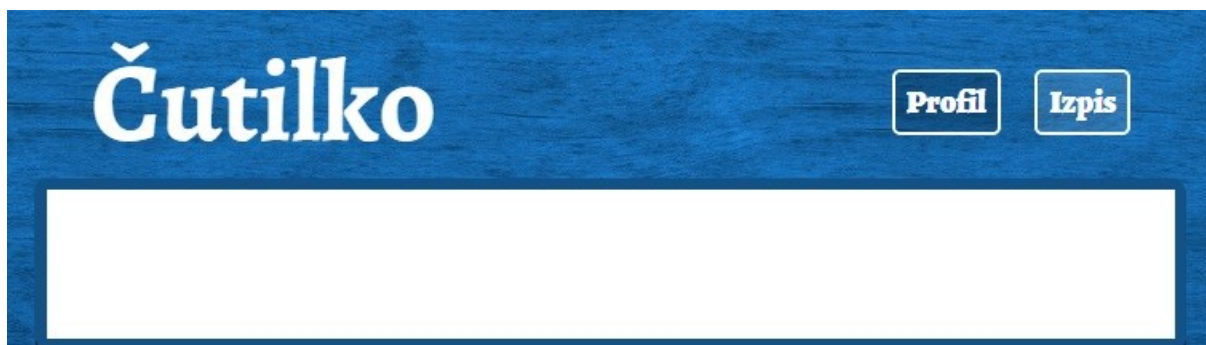
## 5.2 Glavni pogled

Glavni pogled vsebuje kodo HTML, ki je v aplikaciji vedno vidna. Deluje kot okvir za poglede ogrodja AngularJS, ki se znotraj glavnega pogleda izmenjujejo glede na zahteve čutilca. V prvi vrstici kode je definiran modul ogrodja AngularJS, ki naj ga glavni pogled uporablja. Mi smo ustvarili modul `cutilko`, kar definiramo takole: `<html ng-app="cutilko">`. Vse komponente AngularJS, ki so povezane z modulom `cutilko`, so sedaj na voljo za uporabo znotraj kode HTML.

Znotraj glavnega pogleda je definiranih še več stvari:

- Za predstavitev teksta uporablja aplikacija kombinacijo pisav Alegreya, Roboto Condensed in Roboto.
- Poleg po meri napisane kode CSS uporablja aplikacija tudi datoteko `normalize.css`, ki nastavi temeljne stilske lastnosti elementov HTML, kar ponuja konstanten začetek, preden se sami lotimo oblikovanja.
- Vse poti do datotek oziroma skript, ki jih uporablja aplikacija za poganjanje interaktivnosti, so definirane na koncu. To vključuje tudi vse po meri narejene komponente AngularJS.

Glavni del je zelo kratek in vključuje le naslovno vrstico z navigacijskimi gumbi ter belo podlago z robovi, kjer se prikazujejo pogledi ogrodja AngularJS. Slika 5.2.1 prikazuje glavni pogled brez vstavljenega pogleda AngularJS.



Slika 5.2.1: Glavni pogled aplikacije brez vstavljenega pogleda AngularJS.



## Poglavje 6 Odjemalni del

### 6.1 Storitve

AngularJS ima že ob namestitvi na voljo ogromno storitev, ki so pripravljene za takojšnjo uporabo. Tiste, ki smo jih največkrat uporabili, so `$http` za pošiljanje spletnih zahtev, `$state` za upravljanje trenutnega pogleda, `$interval` za delo s časovnimi intervali in `$scope` za delo z objekti in metodami krmilnikov.

Med razvijanjem smo odkrili potrebo po treh dodatnih storitvah:

- Storitev **avten** je namenjena upravljanju žetonov, vpisa, izpisa in registracije čutilcev ter pridobivanju pomembnih informacij o čutilcu.
- Storitev **rezine** deluje kot vmesnik med strežnikom in odjemalcem, ki upravlja z rezinami.
- Storitev **cutilci** deluje kot vmesnik med strežnikom in odjemalcem, ki upravlja s čutilci.

### 6.2 Krmilniki

Med razvijanjem smo ustvarili štiri krmilnike:

- **KrmilnikNavigacije** ja najbolj enostaven. Deluje le kot posrednik med elementi HTML in storitvijo **avten**. Izvorna koda 6.2.1 prikazuje primer enostavnega krmilnika, ki s pomočjo storitve **avten** napolni svoje spremenljivke.

Izvorna koda 6.2.1: Primer enostavnega krmilnika.

```
angular.module('cutilko')
  .controller('KrmilnikNavigacije', ['$scope', 'avten',
    function($scope, avten) {
      $scope.jeVpisan = avten.jeVpisan;
      $scope.izpis = avten.izpis;
      $scope.trenCutilec = avten.trenCutilec;
    }]);
```

- **KrmilnikAvtentikacije** je potreben za delovanje logike vpisa in registracije.
- **KrmilnikRezin** skrbi za programsko logiko prikaza, dodajanja in ustvarjanja rezin na strani odjemalca.
- **KrmilnikProfila** služi upravljanju programske logike delovanja profilne strani. Definirano ima tudi začetno rezino, prikazano na sliki 6.2.1, ki čutilcu razloži strukturo rezin in delovanje aplikacije, vidna pa je samo dokler čutilec ne doda nove, svoje prve rezine.

### To je glavni stavek rezine. Klikni nanj!

Vsaka ideja ima svojo rezino. Sestavljena je iz glavnega stavka, na katerega si ravnokar kliknil, njegov namen pa te je spomniti na pomen ideje. Le-ta je opisan v jedru, ki ga ravnokar bereš, spodnji odstavek pa je namenjen napotkom, ki ti povedo, kako idejo uresničiti v praksi. V desnem zgornjem kotu je ponavadi ime avtorja rezine. Če klikneš nanj, prispeš na njegov javni profil, kjer so prikazane vse rezine, ki jih je ustvaril. Če ti je njegovo delo všeč, lahko pritisneš gumb "Spremljaj" in na svojem profilu boš prejel obvestilo vsakič, ko bo ustvaril novo rezino.

Če greš z miško čez glavni del tvojega profila, se ti pokažeta dva nova gumba. Eden te popelje do zbirke rezin, ki jih še nimaš, a jih lahko dodaš, z drugim pa lahko ustvariš svojo rezino. Vsakič, ko dodaš rezino, ali pa nekdo drug doda rezino, ki si jo ti ustvaril, dobiš točko. Število vseh točk je prikazano pod dosežki, z določenim številom dosežkov pa dobiš novo profilno sliko. Sedaj klikni kjerkoli izven tega okenca, da odideš nazaj na profil. Želimo ti veselo dodajanje, branje in ustvarjanje rezin!

Slika 6.2.1: Začetna rezina, ki čutilcu razloži delovanje aplikacije.

## 6.3 Direktive

Med razvojem se je pokazala potreba po dveh direktivah:

- Direktiva **rezina** je najpomembnejši element celotne aplikacije. Čeprav vsebuje izredno malo kode, ki je prikazana v izvorni kodi 6.3.1, med seboj povezuje programsko logiko iz krmilnika rezin in predlogo `rezina.html`, kar omogoča, da kodo HTML iz predloge uporabimo neštetokrat, ta pa lahko uporablja polno

funkcionalnost krmilnika. Ustvarjamo torej instance predloge HTML, kjer lahko vsaka prikazuje drugačno vsebino, čeprav je koda HTML enaka. To je možno, ker ob izvršitvi posameznim instancam podamo objekte, ki vsebujejo edinstveno vsebino, na primer takole: `<imeDir imeObjvDir=podaniObjekt></imeDir>`.

#### Izvorna koda 6.3.1: Primer enostavne direktive.

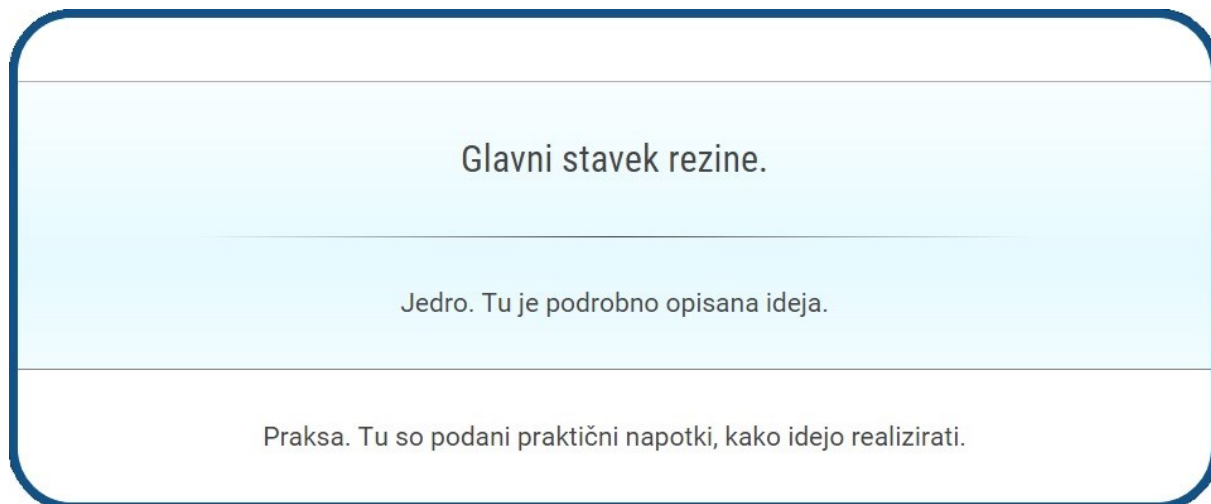
```
angular.module('cutilko')
.directive('rezina', function() {
  return {
    restrict: 'E',
    transclude: true,
    scope: {
      rezina: '='
    },
    templateUrl: 'predloge/rezina.html',
    controller: 'KrmilnikRezin'
  };
});
```

- Direktiva **misel** je nastala zaradi potrebe po spreminjanju besedila elementa `<p>` v kodi HTML. Že obstoječe rešitve nismo našli, zato smo ustvarili svojo. Direktiva upravlja z vrednostmi in prikazovanjem treh elementov HTML, odstavka `<p>`, prostora za pisanje `<textarea>` in gumba `<button>`. Ob spremembi besedila odstavka poskrbi, da se novo besedilo shrani v podatkovno bazo.

## 6.4 Predloga rezina

Predloga je zbirka kode HTML, ki jo uporabljajo direktive, v našem primeru direktiva **rezine**. Bistvo predloge **rezina** je ta, da nam ni potrebno vedno znova pisati kode za koncept rezine. Pričakujemo namreč, da se bo skozi čas v aplikaciji nabralo ogromno rezin, zato je bilo potrebno ustvariti primerno predlogo.

Uporablja krmilnik rezin, ki vsebuje kar nekaj spremenljivk tipa Boolean, ki določajo, kateri elementi rezine so vidni in kateri ne. Rezina je na začetku zaprta, kar pomeni, da je viden le glavni stavek in v primeru, da smo na pogledu za dodajanje rezin, tudi moč. Ko kliknemo na rezino, se le-ta odpre na sredini zaslona, vsebina v ozadju pa izgine, tako da je vidna le odprta rezina. Ta sedaj prikazuje glavni stavek, jedro, prakso in ime avtorja, v primeru, da smo na pogledu za dodajanje rezin, pa še moč in gumb za dodajanje rezine v zbirko čutilčevih dodanih rezin. Na sliki 6.4.1 je vidna struktura rezine.



Slika 6.4.1: Struktura rezine.

## 6.5 Pogledi

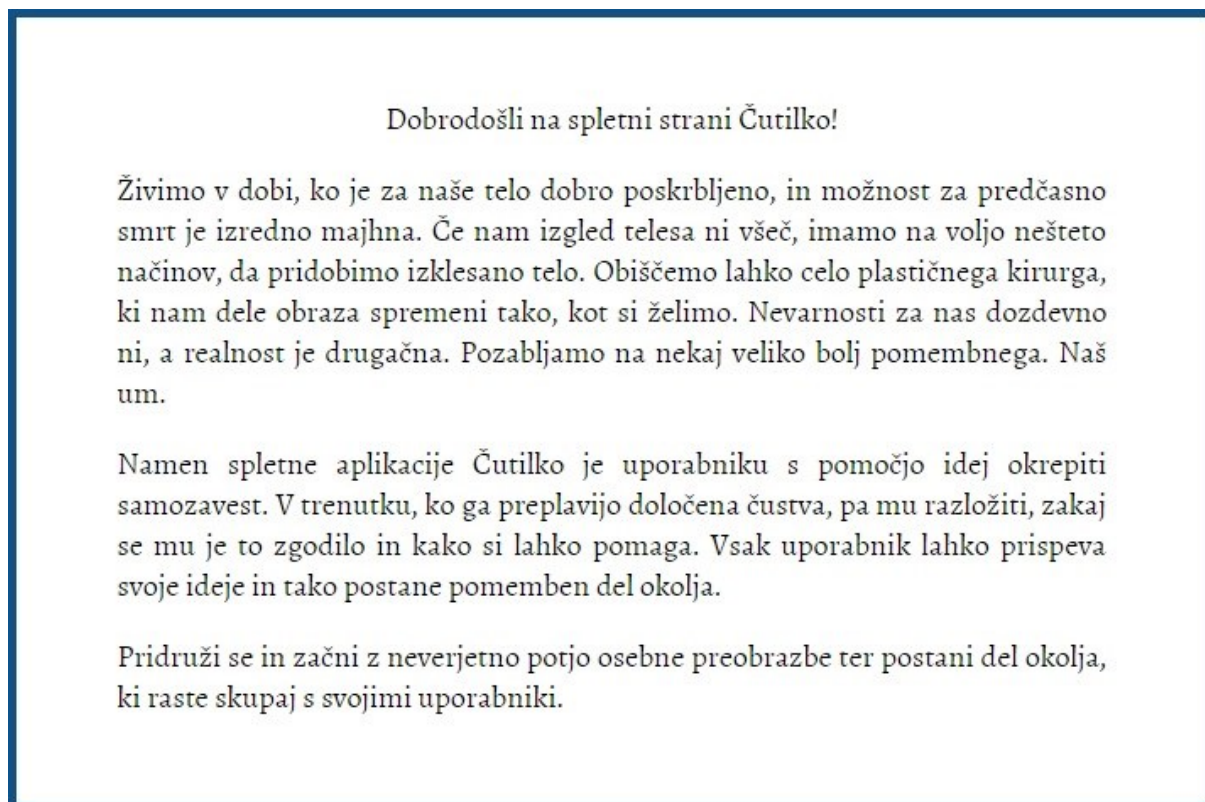
AngularJS uporablja odjemalne poti, ki povedo, kateri pogled naj aplikacija odpre, ko čutilec dostopa do določenega spletnega naslova. Vsi pogledi imajo svojo odjemalno pot z enakim imenom, povezani pa so tudi z določenim krmilnikom. Izvorna koda 6.5.1 prikazuje primer odjemalne poti. Odjemalne poti dopuščajo tudi možnost, da se vsakič, ko se zamenja pogled, izvrši določena koda, ki vnaprej pripravi podatke za ta pogled ali pa na primer preveri, če je čutilec vpisan in sploh ima pravico do vstopa v ta pogled.

Izvorna koda 6.5.1: Primer odjemalne poti.

```
.state('rezine', {
  url: '/rezine',
  templateUrl: '/pogledi/rezine/index.html',
  controller: 'KrmilnikRezin',
  resolve: {
    obljubaRezin: ['rezine', function(rezine) {
      return rezine.dobiRezine();
    }]
  },
  onEnter: ['$state', 'avten', function($state, avten) {
    {
      if (!avten.jeVpisan()) {
        $state.go('vstopna');
      }
    }
  }]
})
```

### 6.5.1 Vstopni pogled

Vstopni pogled, prikazan na sliki 6.5.1.1, je viden vsem, ki obišejo spletno aplikacijo, in ne uporablja nobenih drugih funkcionalnosti ogrodja AngularJS. Vsebuje stavek za pozdrav in nekaj odstavkov, ki dajo obiskovalcu idejo o tem, čemu aplikacija služi. To se nam zdi pomembno, saj smo mnenja, da se bo redko kdo registriral v aplikaciji, katere namen ne pozna.



Slika 6.5.1.1: Vstopni pogled.

### 6.5.2 Pogleda za vpis in registracijo

Pogleda za vpis in registracijo, prikazana na slikah 6.5.2.1 in 6.5.2.2, sta povezana s krmilnikom avtentikacije. Ta vsebuje prazen objekt `cutilec`, ki se napolni z vrednostmi, pridobljenimi iz spletnih obrazcev. Ko čutilec izpolni polji za ime in geslo čutilca ter pritisne na gumb za vpis oziroma registracijo, se najprej izvrši metoda `velikaZacetnica(ime)`, ki spremeni podano ime v format, kjer je prva črka velika črka, vse druge pa male. Za tem se izvrši še glavna metoda `vpis()` oziroma `registracija()`. Ta izvrši enako poimenovano metodo iz storitve **avten** ter zraven kot argument poda objekt `cutilec`, kar dejansko vpiše oziroma registrira čutilca ter ga preusmeri na pogled njegovega profila. Če se

pri procesu pojavi težava, se v krmilniku ustvari nova spremenljivka napaka, ki se uporabi pri obveščanju čutilca o neuspehi avtentikaciji.

Polje za pisanje imena je omejeno na minimalno tri in maksimalno pet črk ter ob praznem polju prikazuje besedilo »Ime Čutilca«. Polje za pisanje gesla nima posebnih omejitev, ob praznem polju pa prikazuje besedilo »Geslo«. V primeru, da čutilec še ni registriran, ima v pogledu za vpis na voljo povezavo do pogleda za registracijo.



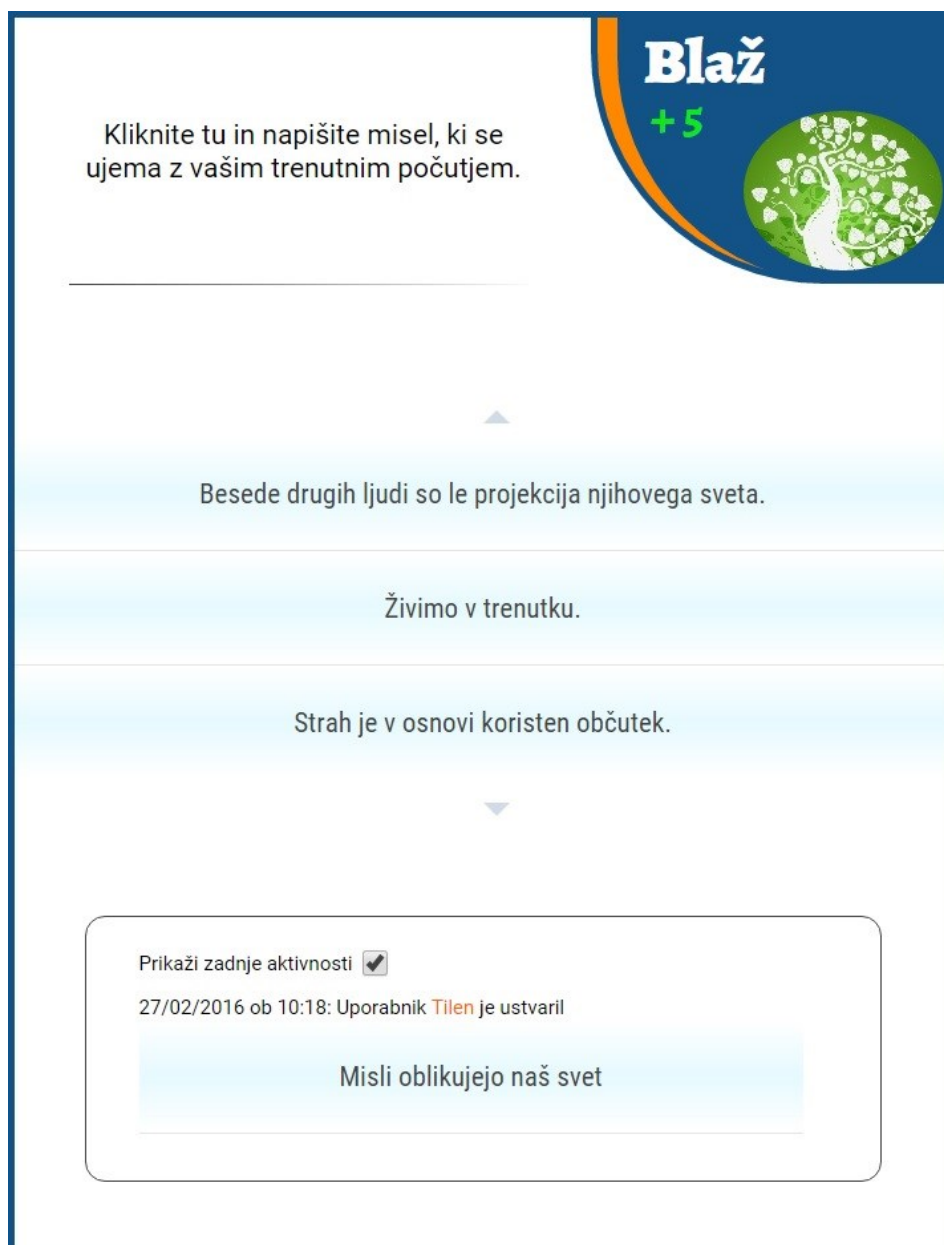
Slika 6.5.2.1: Obrazec za vpis.



Slika 6.5.2.2: Obrazec za registracijo.

### 6.5.3 Pogled profila

Ob preklopu na pogled profila, ki je povezan s krmilnikom profila, se najprej izvrši `obljubaProfila`. Ta izvrši metodo `dobiProfil(null)` iz storitve **cutilci**, ki pošlje GET zahtevo, s katero strežnik vrne vse podatke trenutnega čutilca. Krmilnik profila nato uporabi te podatke, da napolni svoje spremenljivke. Če obiščemo profil sočutilca, se namesto parametra `null` uporabi ključ sočutilca, kar izvrši drugo kodo metode in vrne podatke za sočutilca. Slika 6.5.3.1 prikazuje primer pogleda profila čutilca.



Slika 6.5.3.1: Primer pogleda profila čutilca.

Profil je sestavljen iz štirih glavnih elementov:

- **Misel** zaseda približno 60% prve vrste elementov in je instanca direktive **misel**. Odstavek znotraj direktive prikazuje vrednost spremenljivke `profil.misel` iz krmilnika profila. Ob kliku nanj se odstavek skrije, prikaže pa se polje za urejanje besedila odstavka ter gumb za shranjevanje. Ko besedilo spremenimo in kliknemo na gumb, se izvrši metoda `spmeniMisel()` iz storitve **cutilci**, ki pošlje PUT zahtevo skupaj z novo mislijo, ki v podatkovni bazi zamenja staro.

- **Profilka** zaseda preostali del prve vrste elementov in prikazuje ime čutilca, njegove dosežke in pa sliko profila. S pomočjo CSSa smo ustvarili modro podlago in pa oranžno krivuljo, ki poudari mejo med profilko in ostalimi elementi, kot je to vidno na sliki 6.5.3.1 zgoraj desno. Dosežki so napisani v zeleni barvi, ker želimo, da predstavljajo nekaj pozitivnega. Slika za svoj vir (atribut `src` v znački `<img>`) ne uporablja direktno niza, temveč uporablja niz, ki ga vrne metoda `dobiSliko()`. Ta preveri število dosežkov čutilca in za določeno število vrne niz lokacije določene slike. S tem čutilec dobiva vedno bolj navdušujoče slike.
- **Jedro profila** zaseda celotno naslednjo vrsto elementov. V krmilniku profila je tabela `pogledRezin`, ki dobi tri rezine iz tabele `profil.dodano` in jih s pomočjo direktive **rezine** prikazuje na profilu. Te rezine se izmenjujejo na več načinov. Krmilnik ima metodi `gor()` in `dol()`, ki tabelo `pogledRezin` napolnita z novimi največ tremi rezinami, ki so v tabeli `profil.dodano` na enem indeksu nižje oziroma višje, odvisno od metode. Če kliknemo na gumb nad jedrom profila, se izvrši prva, če kliknemo na gumb pod jedrom pa druga metoda. Gumba sta aktivna le, ko imamo dodane več kot tri rezine. V kolikor se ne vmešavamo, se s pomočjo intervala vsake pol minute samodejno požene metoda `dol()`. Ko se z miško pomaknemo čez jedro profila, se prikažeta tudi gumba za preklop na poglede dodajanja in ustvarjanja rezin.
- **Zgodovina** zaseda celotno zadnjo vrsto elementov in je prikazana le, če tabela `profil.zgodovina` ni prazna. Če je temu tako, lahko s pomočjo gumba določimo, ali bomo podatke prikazovali. Tabela vsebuje zadnje aktivnosti vseh sočutilcev, ki jih spremljamo. Aktivnost vsebuje čas, ko je bila aktivnost zabeležena, ime sočutilca, na katerega se navezuje, kakšno je dejanje sočutilca in katera rezina je z aktivnostjo povezana. Urejene so po času padajoče, prikazuje se le zadnjih pet, rezine pa so znova predstavljene z direktivo **rezine**. Če kliknemo na ime sočutilca, nas to popelje na njegovo profilno stran.

#### 6.5.4 Pogled rezin

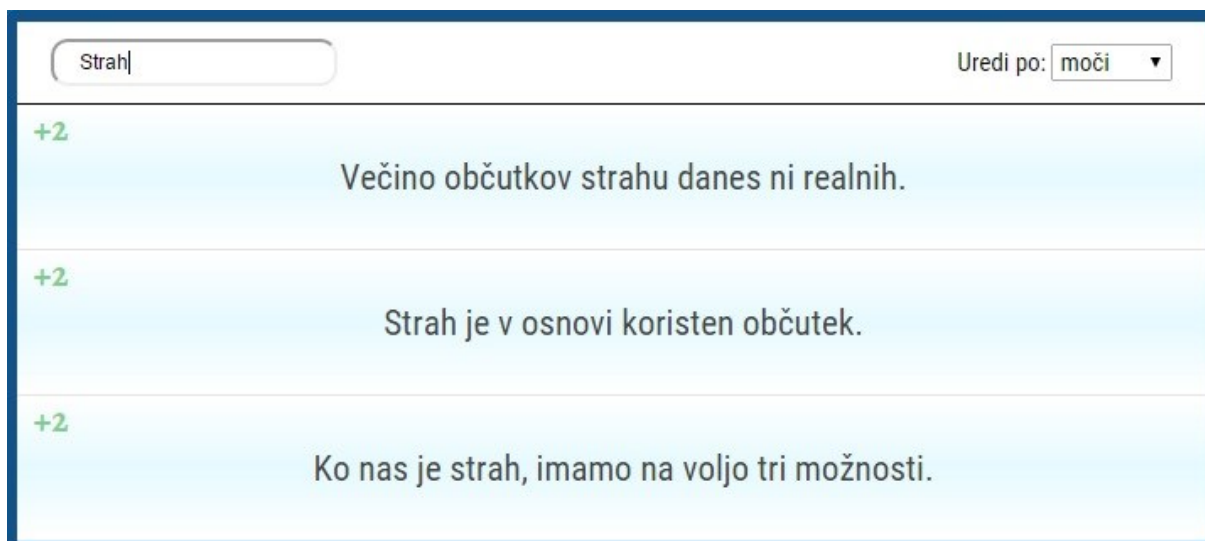
Ob preklopu na pogled rezin, ki je povezan s krmilnikom rezin, se najprej izvrši `obljubaRezin`. Ta izvrši metodo `dobiRezine()` iz storitve **rezine**, ki pošlje GET zahtevo, kot je prikazano v izvorni kodi 6.5.4.1, s katero strežnik vrne vse rezine, ki jih čutilec še nima v zbirki dodanih rezin. Lastnost `headers` je uporabljena za preverjanje pristnosti čutilca na strani strežnika.



## Izvorna koda 6.5.4.1: Primer \$http zahteve GET.

```
r.dobiRezine = function() {  
    return $http.get('/api/v1/rezine/' + avten.idTrenCu  
    tilca(), {  
        headers: {  
            Authorization: 'Bearer ' + avten.dobiZeton()  
        }  
    }).success(function(rezine) {  
        angular.copy(rezine, r.rezine);  
    });  
};
```

Pogled, prikazan na sliki 6.5.4.1, ima na vrhu polje, kjer lahko iščemo rezine, ki vsebujejo določene besede, in meni, kjer izbiramo med urejenostjo rezin glede na moč ali čas nastanka. Od tu dalje so prikazane vse rezine, ki jih uporabnik še nima dodanih. Vidita se le moč in glavni stavek rezine, kar pomeni, da so rezine zaprte. Če katerikoli stavek v nas vzbudi zanimanje, lahko nanj kliknemo in rezina se odpre. To se zgodi z izvršitvijo metode `odpri()`, ki nastavi določeno Boolean spremenljivko na vrednost `true`, kar povzroči, da se rezini spremeni stil prikaza. Vidni postanejo še jedro, praksa, ime avtorja in pa gumb za dodajanje. Ob kliku na ime avtorja preklapimo na njegov profil, ob kliku na gumb za dodajanje pa se izvrši metoda `dodajRezino(rezina)`, ki rezino doda v zbirko dodanih rezin ter jo odstrani iz seznama ne dodanih. Ob kliku v prazno polje poleg odprte rezine se le-ta zapre.



Slika 6.5.4.1: Pogled za dodajanje rezin.

### 6.5.5 Pogled ustvarjanja rezin

Pogled ustvari, prikazan na sliki 3.1.2, s pomočjo obrazca napolni objekt `novaRezina` s tremi lastnostmi:

1. V prvo polje vpišemo glavni stavek ideje, ki napolni lastnost `novaRezina.stavek`. Minimalna dolžina besedila je deset črk, maksimalna pa petdeset. Ko je polje prazno je izpisano besedilo »Napišite glavni stavek ideje...«.
2. V drugo polje vpišemo jedro rezine, ki napolni lastnost `novaRezina.jedro`. Minimalna dolžina besedila je sto črk, maksimalna pa petsto. Ko je polje prazno je izpisano besedilo »Na široko opišite idejo...«.
3. V zadnjem polju vpišemo prakso rezine, ki napolni lastnost `novaRezina.praksa`. Minimalna dolžina besedila je petdeset črk, maksimalna pa tristo. Ko je polje prazno je izpisano besedilo »Podajte napotke, kako idejo realizirati v praksi...«.

Vsa tri polja morajo biti obvezno izpolnjena, da se rezina lahko doda. Ko kliknemo na gumb »Ustvari rezino«, se izvrši metoda `ustvariRezino()` iz krmilnika `rezin`, ki objektu `novaRezina` doda še ključ in ime trenutnega čutilca. Za tem uporabi enako poimenovano metodo storitve **rezina** ter tako ustvari novo rezino v podatkovni bazi. Če rezine ne želimo ustvariti, lahko kliknemo gumb, ki nas vrne nazaj na pogled profila.

### 6.5.6 Pogled profila sočutilca

Pogled profila sočutilca je prikazan na sliki 6.5.6.1 in je skoraj popolnoma enak pogledu profila, le prirejen je za sočutilca. Odstranjena je možnost spreminjanja misli in izpis zgodovine oziroma zadnjih aktivnosti. Prav tako se v jedru profila ne izpisujejo dodane rezine sočutilca, ampak tiste, ki jih je ustvaril. Dodan je gumb »Spremljaj«, ki izvrši metodo `dodajSocutilca()`, ki sočutilca doda v zbirko sočutilcev, ki jih čutilec spremlja.



Slika 6.5.6.1: Profilna stran sočutilca.

## 6.6 Osnovne rezine iz aplikacije

Osnovne rezine, ki smo jih vključili v aplikacijo, so:

1. Besede drugih ljudi so le projekcija njihovega sveta: slika 6.6.1.
2. Če nas nekdo namenoma ponižuje, za to nismo krivi mi: slika 6.6.2.
3. Misli oblikujejo naš svet: slika 6.6.3.
4. Zaupati moramo lastni presoji in na stvari gledati z lastnimi očmi: slika 6.6.4.
5. Bodimo glavna zvezda našega življenja: slika 6.6.5.
6. Živimo v trenutku: slika 6.6.6.
7. Pohvala nima vedno željenega učinka: slika 6.6.7.
8. Fizično smo enakovredni drugim: slika 6.6.8.
9. Značaj se oblikuje skozi odraščanje: slika 6.6.9.
10. Ljudje se lahko ustrašijo samozavesti: slika 6.6.10.
11. Ne postanimo odvisni od tehnologije: slika 6.6.11.
12. Materialne dobrine ne zapolnijo praznine: slika 6.6.12.
13. Ideja dobi pravo vrednost šele takrat, ko jo pričnemo uresničevati: slika 6.6.13.
14. Brez zadržkov izražajmo svojo osebnost in ne glejmo na posledice: slika 6.6.14.
15. Ne bojimo se delati napak: slika 6.6.15.
16. Sposobni moramo biti živeti sami: slika 6.6.16.
17. Od sebe ne pričakujmo popolnosti: slika 6.6.17.
18. Večina strahov danes ni realnih: slika 6.6.18.
19. Strah je v osnovi koristen občutek: slika 6.6.19.
20. Ko nas je strah, imamo na voljo tri možnosti: slika 6.6.20.
21. Občutek strahu je tak z razlogom: slika 6.6.21.
22. Dihanje je kratek stik za paniko: slika 6.6.22.
23. V trenutku panike uporabimo pomirjevalni pristop: slika 6.6.23.

### Besede drugih ljudi so le projekcija njihovega sveta.

Besede drugih ljudi ne smemo jemati osebno. So le verbalno izražanje sveta, v katerem živijo. Vsak človek živi v svojem svetu, s svojim mišljenjem in svojimi pravili, ki niso nujno enaka našim. Niso vse reči podkovane z dejstvi, zato je resnica to, za kar se posameznik odloči verjeti.

Ko slišimo neke besede, se lahko odločimo, ali v njih verjamemo ali ne. Odločitev je naša. Če se z mišljenjem nekoga ne strinjamo, imamo vso pravico obdržati svoje mnenje in osebo ignorirati, če nam svoje stališče želi vsiliti. Tvoja resnica je to, v kar ti verjameš.

Slika 6.6.1: Besede drugih ljudi so le projekcija njihovega sveta.

### Če nas nekdo namenoma ponižuje, za to nismo krivi mi.

Ljudje smo rojeni dobrega srca in nihče s takim srcem ne bi drugemu človeku želel hudega. Tisti, ki čutijo potrebo po žaljenju in poniževanju, so globoko v sebi nezadovoljni sami s seboj. V njihovem svetu je nekaj negativnega, kar podzavestno izražajo s svojimi besedami. Druge ponižujejo, ker jih želijo spraviti na svojo raven, in ne zato, ker bi bilo z njimi dejansko nekaj narobe.

Ko nas nekdo želi ponižati, tega ne jemljimo osebno. Vprašajmo se, kaj v življenju ga je privedlo do tega, da se počuti drugim manj vrednega. Če človeka s tem soočimo, verjetno tega ne bo sprejel najbolje, a mu bo mogoče dalo misliti, da je težava dejansko pri njem.

Slika 6.6.2: Če nas nekdo namenoma ponižuje, za to nismo krivi mi.

### Misli oblikujejo naš svet.

Roža sama po sebi ni ne lepa ne grda. To so lastnosti, ki jih dodelimo ljudje sami, glede na naše dožemanje sveta. Resnica je, da obstaja. Roža se ne počuti lepo ali grdo, ampak enostavno je. To velja za vse stvari na svetu razen ljudi, ki si svet z mislimi oblikujemo sami. Če smo večino časa pozitivni, je ogromna možnost, da bomo nekaj doželi kot tako, če pa smo negativni, se nam lahko tudi najlepša roža zdi grda.

Ko se nam nekaj zdi grdo, premislimo, ali je razlog naš negativni pogled na svet. Ustavimo se in poizkusimo stvari videti kot resnico brez predsodkov. Zavedimo se, da jih oblikujemo sami z našimi mislimi. Takrat se potrudimo razmišljati pozitivno in oblikujemo svoj svet na novo. Prej grda reč nam bo sedaj morda izgledala drugače.

Slika 6.6.3: Misli oblikujejo naš svet.

### Zaupati moramo lastni presoji in na stvari gledati z lastnimi očmi.

Že od malih nog smo izpostavljeni družbenemu pogojevanju. Starši in učitelji nas učijo kaj se sme in kaj ne. Ne igravimo se z ognjem, drugače se bomo spekli. Hodimo v šolo, drugače v življenju ne bomo uspešni. Družbeno pogojevanje je za nas zelo koristno. Omogoča nam, da se izredno hitro naučimo stvari, ki so pomembne za preživetje, vendar ima tudi slabo plat. Ideje, vseeno kako neumne, se lahko nenadzorovano širijo. Še ne dolgo nazaj smo bili prepričani, da je zemlja ploščata. Ne smemo brezglavo verjeti vse, kar nam je rečeno. Zaupati moramo samemu sebi.

Dvomimo v vse, česar sami ne potrdimo. Sprašujmo se, zakaj so stvari takšne kot so. Zakaj so hiše pravokotne oblike? Zakaj ne smemo čez cesto, ko je na semaforju rdeča luč? Zakaj ni dobro, da smo z ljudmi odkriti, če jih s tem užalimo? Sami pridimo do zaključka in naj nam ne bo nerodno, če naš pogled ne ustreza družbenim normam.

Slika 6.6.4: Zaupati moramo lastni presoji in na stvari gledati z lastnimi očmi[18].



### Bodimo glavna zvezda našega življenja.

Vse preveč le opazujemo druge ljudi in občudujemo njihove spretnosti in dosežke. Kujemo jih v zvezde, medtem ko od sebe ne pričakujemo veliko, ali pa ne verjamemo, da smo tega zmožni tudi sami. Želimo si, da bi bili taki kot naši vzorniki, a smo prevečkrat le opazovalci. Vsi imamo svoje življenje, svoj film, ki ga doživljamo sami na svoj edinstven način, a sebe premalokrat postavimo v glavno vlogo. To prepuščamo drugim, sami pa smo raje statist. Ko umremo, je našega filma na zemlji konec, zato poskrbimo, da si ga bo drugim vredno ogledati še enkrat.

Ko si želimo nekaj storiti, vendar le opazujemo ljudi, ki to počnejo izvrstno, si predstavljamo dvorano v kinu. Vrtijo naš film in mi smo edini gledalec, a film ne prikazuje našega življenja. Prikazuje neko drugo dvorano, ki je polna, vrti pa se film človeka, ki ga opazujemo. Tako je lahko naše življenje, če prepustimo, da v našem filmu izgubimo glavno vlogo. Stopimo v to vlogo in sami določimo potek našega filma.

Slika 6.6.5: Bodimo glavna zvezda našega življenja.

### Živimo v trenutku.

Preteklosti ne moremo spremeniti, prihodnosti pa ne napovedati. Večino nas vseeno bremenijo dogodki iz preteklosti in skrbi nas, kaj bo prišlo v prihodnosti. Edino, na kar lahko vplivamo, je kar storimo in čutimo v tem trenutku.

Ne obremenjujmo se z rečmi, ki so izven našega dosega. Potrudimo se, da nam misli ne uhajajo na pretekle dogodke in da ne pričakujemo prihodnosti preden se zgodi. Osredotočimo se na stvari, ki se dogajajo v tem trenutku. Ustavimo se in si oglejmo okolico. Kje smo, kaj se dogaja v tem trenutku in kaj lahko storimo sedaj?

Slika 6.6.6: Živimo v trenutku.

### Pohvala nima vedno željenega učinka.

Prejete pohvale se lahko hitro spremenijo v lastna pričakovanja. Mame otrokom že od rojstva govorijo, na primer, kako so lepi ali pametni. S tako idejo ti otroci odrastejo, od sebe pa še vedno pričakujejo, da so taki. V življenju smo hitro postavljeni na trdna tla, ko kar naenkrat ne izpolnjujemo več svojih nerealnih pričakovanj. To lahko močno vpliva na našo samozavest.

Ko želimo nekoga pohvaliti, namesto, da rečemo: "Ti si pa zelo pameten.", raje recimo: "Tega si se pa zelo pametno domislil". Namesto, da rečemo: "Kako lepo plešeš.", recimo: "Kako lepo si tokrat odplesal.". S tem človeka pohvalimo za nekaj, kar je storil, in mu ne dodeljemo neko lastnost, ki jo spremeni v nepotrebna pričakovanja.

Slika 6.6.7: Pohvala nima vedno željenega učinka[1].

### Fizično smo enakovredni drugim.

Dva zdrava človeka sta si fizično enakovredna. Oba imata dve roki, usta, nos, oči in ušesa. Oba sta zmožna govoriti in komunicirati. Fizično smo sposobni delati in govoriti enake stvari, kot jih dela in govori nekdo, ki ga imamo za nekaj več. Edina razlika je v miselnosti. Če smo samozavestni in verjamemo, da lahko enako počnemo tudi mi, ostane malo stvari, ki nam to lahko preprečijo.

Če imamo v življenju nekoga, ki ga občudujemo in ga imamo za nekaj več, bodimo pozorni na njegovo gibanje, govorjenje in način komuniciranja. Poizkusimo to ponoviti pred ogledalom, ko okoli nas ni nikogar. Je to res nekaj, česar sami ne zmoremo?

Slika 6.6.8: Fizično smo enakovredni drugim.



### Značaj se oblikuje skozi odraščanje.

Z značajem nismo rojeni, ampak ga gradimo z izkušnjami med odraščanjem. Če so nam starši dajali vse, kar smo si želeli, danes morda težko sprejmemo besedo ne. Če so nas okregali, ko smo si nekaj zaželeli, se morda danes to, kar si želimo, bojimo zahtevati. Res, da smo z nekaterimi lastnostmi rojeni, a večino stvari smo se naučili in to kar se naučimo, lahko spremenimo.

Ko imamo občutek, da bi radi nekaj storili, pa nam to ni podobno, zberimo vso voljo, ki jo imamo, in naredimo korak k spremembi. To storimo vsakokrat, ko se znajdemo v podobnem položaju. Z vztrajnostjo bomo spremenili svoj značaj, a bodimo pozorni. Če imamo preveč negativnih misli, lahko tudi te postanejo del našega značaja, zato poskrbimo, da so naše misli vedno pozitivne.

Slika 6.6.9: Značaj se oblikuje skozi odraščanje[1].

### Ljudje se lahko ustrašijo samozavesti.

Prava samozavest danes ni zelo pogosta, zato se je ljudje lahko ustrašijo. Ko smo samozavestni, oddajamo energijo, ki druge ljudi lahko preplavi. Takrat podvomijo vase, ne vedo, kako se s položajem soočiti in se v skrajni sili celo umaknejo. Nam to lahko izpade, kot da to osebo ne zanimamo, kar lahko vpliva tudi na našo samozavest.

Če neka oseba ne kaže zanimanja, ne predpostavimo takoj, da se ne želi pogovarjati z nami, temveč raje predpostavimo, da ni dovolj samozavestna. Osebi, ki nam je všeč, a se nas zaradi naše samozavesti boji, moramo pokazati, da ji je z nami lahko udobno. Najbolje je, da ji razkrijemo nekaj osebnega, ali pa jo z nečim pohvalimo in ji tako sporočimo, da ima za nas neko vrednost, zaradi česar se bo lažje odprla.

Slika 6.6.10: Ljudje se lahko ustrašijo samozavesti.

### Ne postanimo odvisni od tehnologije.

Danes se je težko izogniti tehnologiji. Televizije, računalniki in telefoni so neskončen vir zabave, informacij in stimulacije, od katerih zlahka postanemo odvisni. V skrajnem primeru začnemo zanemarjati svoje telo in um, v zameno za življenje, ki ga doživljamo skozi ekran. Zaradi tega lahko trpijo vsi vidiki našega življenja. Šele, ko se te odvisnosti znebimo, se zavemo, da si življenje, ki ga vidimo v ekranu, lahko ustvarimo sami.

Ne dovolimo, da postanemo odvisni od tehnologije. Nič nam ne prepričuje, da sami počnemo stvari, ki jih vidimo v ekranu. Odložimo telefon, ugasnimo televizijo in izklopimo računalnik. Predstavljajmo si, da smo mi tisti, ki ga drugi vidijo v svojem ekranu. Ustvarimo si svoj zabaven svet.

Slika 6.6.11: Ne postanimo odvisni od tehnologije.

### Materialne dobrine ne zapolnijo praznine.

Pogosto, ko kupimo nekaj novega, na primer čevlje, telefon ali oblačila, dobimo občutek sreče, ki traja nekaj časa, potem pa si spet želimo nekaj drugega. Materialne reči nas nikoli zares ne zapolnijo tako, kot nas lahko zapolnijo druženja s prijatelji, hoja po naravi, doseganje ciljev itd. Kupljeni predmeti imajo za nas neko vrednost, ki nam daje občutek, da smo s tem nekaj več vredni tudi sami. Ta občutek je seveda prazen, naša prava vrednost pa ostaja skrita. Vsi jo imamo, le poiskati jo moramo.

Ko imamo občutek praznine, ne iščimo vrednost v drugih stvareh. Verjemimo, da smo z vrednostjo že rojeni, le izgubili smo ta občutek. Odkriti moramo, kako smo lahko srečni in pozitivni brez tujega vpliva. Obiščimo prijatelje, sprehodimo se po naravi, opazujemo živali, skrbimo za naše telo. Ko smo srečni in pozitivni brez vpliva materialnih dobrin, smo že našli svojo vrednost.

Slika 6.6.12: Materialne dobrine ne zapolnijo praznine[1].

### Ideja dobi pravo vrednost šele takrat, ko jo pričnemo uresničevati.

Ideja brez udejstvovanja nima vrednosti. Lahko si jo ponovimo tisočkrat, pa bo še vedno le misel v naši glavi. Ko jo pričnemo uresničevati, ta pridobiva otipljivo vrednost, ki vedno bolj vpliva na naš svet. Če idejo uresničujemo dovolj časa, postane del nas ali pa fizičen del sveta, v katerem živimo. Ideja, ki se ne obnese takoj, ni nujno slaba, in včasih je potrebno ogromno volje, da se uresniči.

Ko dobimo idejo, si jo zapišemo na list papirja, ter se k njej vrnemo, ko imamo čas. Čimprej jo začnemo uresničevati, dokler ta ne dobi otipljive vrednosti, postane del nas ali pa fizičen del sveta. Potrudimo se, da ne odnehamo ob prvem znaku težav, temveč v idejo verjamemo do konca.

Slika 6.6.13: Ideja dobi pravo vrednost šele takrat, ko jo pričnemo uresničevati[18].

### Brez zadržkov izražajmo svojo osebnost in ne glejmo na posledice.

Ljudje hitro ugotovijo, kdaj izbiramo in prirejamo svoje besede, da bi na njih naredili dober vtis. Kot takim nam bodo težko zaupali. Bolje je povedati to, kar si mislimo, pa čeprav izpademo neumni. Ko prirejamo naše besede, ne moremo biti spontani, zaradi česar se lahko oseba, s katero govorimo, počuti nelagodno. Govorjenje brez zadržkov pokaže, da zaupamo v to kar smo, in da to lahko podpremo z besedami.

Ko začnemo prirejati svoje besede, se spomnimo, da bo dober vtis nekaj vreden le, če smo s tem pokazali to, kar smo. Lažen dober prvi vtis je nekaj, kar moramo v prihodnosti tudi ohranjati, drugače bo oseba prepoznala resnico in izgubila zaupanje v nas. Bodimo to kar smo, četudi smo mnenja, da bomo nekoga odgnali. Če se to zgodi, pomeni, da ni bila prava oseba za nas.

Slika 6.6.14: Brez zadržkov izražajmo svojo osebnost in ne glejmo na posledice[19].



### Ne bojmo se delati napak.

Napake so pomembne za nas in iz njih se učimo že celo življenje. Nihče ni prvič sedel na kolo in se znal brezhibno peljati brez rok. To se je naučil iz napak in verjetno zelo bolečih. Brez njih ne bi napredovali. Vsakič, ko naredimo napako, se iz nje nekaj naučimo in izboljšamo del sebe. Vsaka napaka je pomembna izkušnja, s katero rastemo kot oseba.

Ko storimo napako, se ne počutimo osramočenega ali ponižanega, saj smo dosegli ravno nasprotno. Iz te izkušnje smo se naučili nekaj pomembnega in del nas je sedaj še kanček boljši, kot je bil prej.

Slika 6.6.15: Ne bojmo se delati napak[1].

### Sposobni moramo biti živeti sami.

Življenje s partnerjem je lahko čudovito, vendar le če sta oba partnerja sposobna poskrbeti zase in v pravem času biti samostojna. Tako si lahko postavita meje, ki jih oba potrebujeta, in sklepata kompromise glede njunega skupnega življenja. Če je eden od partnerjev šibek, ta lahko za lastno preživetje postane odvisen od drugega. Nanj se začne zanašati za odgovornost in močno voljo, sam pa te sposobnosti izgublja. Lahko se zgodi, da samostojni partner izgubi privlačnost do šibkega ter ga zapusti, ta pa brez njega ne more več živeti.

Poskrbimo, da imamo s partnerjem vzpostavljene trdne meje in naložene vsak svoje odgovornosti. Postavimo se za svoja načela, a bodimo pripravljeni sprejemati kompromise. Ne dovolimo pa, da postanemo odvisni od partnerja ali obratno, saj bo to škodilo nam in razmerju.

Slika 6.6.16: Sposobni moramo biti živeti sami[20].

### Od sebe ne pričakujmo popolnosti.

Popolnost človeka bi pomenila, da obstaja le en pravi način za doseg cilja, le en pravi izgled. Vsak človek je že rojen poseben in ko odraste mu ni nihče drug podoben. Popolnost je izmišljena vrlina ljudi, ki mislijo, da je njihov način najboljši. Nihče na svetu nima pravice trditi tega, ko pa smo si v osnovi vsi drugačni.

Sprejmimo se takšne kot smo. Smo edinstveni in nihče drug nam ni podoben. Dobro je, če želimo postati boljši, vendar postanimo boljša različica samih sebe, ne pa različica nekoga drugega.

Slika 6.6.17: Od sebe ne pričakujmo popolnosti[1].

### Večina strahov danes ni realnih.

Živimo v času, ko je za fizično varnost človeka dobro poskrbljeno in strah, da lahko umreš za vsakih vogalom, ni več prisoten, smo pa se zato naučili novih strahov. Strah pred neuspehom, strah pred javnim govorom, strah pred tem, da bomo koga razočarali ali užalili so le nekateri izmed njih. Ne bojimo se več, kaj se lahko zgodi našemu telesu, temveč kaj se lahko zgodi našemu umu. Um pa je veliko težje zaceliti. Ti novodobni strahovi so neutemeljeni in ne realni. Zaradi neuspeha ne bomo umrli, prav tako nas ne bodo pretepli ali izločili iz skupnosti, če na nekoga ne naredimo dober vtis.

Ko nas je strah, dobro premislimo, zakaj je tako. Smo v smrtni nevarnosti? Verjetno ne. Čemu nas je potem strah? Kaj je najhujše, kar se lahko zgodi? Bomo pretepeni? Bomo izobčeni iz družbe? Bomo izgubili vse, kar nam je ljubo? Česa se sploh bojimo?

Slika 6.6.18: Večina strahov danes ni realnih[21].

### Strah je v osnovi koristen občutek.

Namen strahu je, da se izognemo grožnjam, ki nas lahko poškodujejo ali celo ubijejo. Ko plavamo v morju in zagledamo morskega psa, imamo od strahu izredno korist. Kar se da hitro zbežimo nazaj na kopno, stran od nevarnosti. Brez strahu bi torej težko živeli. Dvomljiva pa je korist strahu v primeru, ko se bojimo stvari, ki nam fizično niso nevarne. Kakšna je korist strahu, ko moramo na primer javno govoriti ali pa govorimo z osebo, s katero nam ni lagodno? Od bežanja tokrat nimamo nobene koristi, obratno, ovira našo sposobnost opravljanja vsakdanjih opravil.

Ko nas je strah, premislimo, kaj je bolj koristno. Je beg res tisto, kar nas bo rešilo nevarnosti, ali nevarnosti v resnici ni, in nas strah le ovira pri doseganju naših ciljev?

Slika 6.6.19: Strah je v osnovi koristen občutek[21].

### Ko nas je strah, imamo na voljo tri možnosti.

Lahko zamrznemo in se s tem skušamo skriti pred nevarnostjo, zbežimo in upamo, da nas nevarnost ne ujame, ali pa se borimo in upamo, da nevarnost premagamo. Katera možnost je najboljša je v tistem trenutku težko določiti, saj ne razmišljamo jasno, a v svetu, kjer je strah v naših glavah večinoma izmišljen, bi se morali z nevarnostjo večkrat boriti.

Potrudimo se, da se z nevarnostjo čim večkrat borimo, oziroma spoprimemo. Sčasoma bomo ugotovili, da strah največkrat nima trdnih temeljev in je le miselna prepreka na naši poti.

Slika 6.6.20: Ko nas je strah, imamo na voljo tri možnosti[21].

### Občutek strahu je tak z razlogom.

Ko nas je strah, telo misli, da smo v nevarnosti, zato sproži varnostne mehanizme. Ne razmišljamo več jasno, ampak impulzivno, zato je komuniciranje oteženo. Poveča se nam srčni utrip, krvni pritisk, raven adrenalina v krvi ter potreba po kisiku. Telo je v tem stanju najbolj zmožno bežati ali pa se boriti z nevarnostjo.

Ko nas je strah, bodimo pozorni na svoje telo in um. Če simptomov strahu ne razumemo, to lahko povzroči še večjo paniko, zato se navadimo na to, kako se naše telo obnaša v primeru nevarnosti.

Slika 6.6.21: Občutek strahu je tak z razlogom[21].

### Dihanje je kratek stik za paniko.

Hitro, plitvo dihanje, je prvi sprožilec, ki zažene vse druge simptome panike, zato lahko z nadzorovanjem dihanja nadzorujemo tudi vse druge simptome panike. Če namenoma izdihujemo dlje, kot pa vdihujemo, telo nima druge izbire, kot da se umiri.

Ko nas začenja grabiti panika se ustavimo se. Osredotočimo se na dihanje. Vdihujemo ob hitrem štetju do 7 (v mislih) in izdihujemo ob hitrem štetju do 11. Če to počnemo kakšno minuto, bomo presenečeni, kako hitro smo se umirili. Temu rečemo "7/11 dihanje", a številke so lahko poljubne, le izdihovanje mora biti daljše od vdihovanja.

Slika 6.6.22: Dihanje je kratek stik za paniko[22].



### V trenutku panike uporabimo pomirjevalni pristop.

Strah in panika dajeta občutek, kot da se nam enostavno zgodita, vendar imamo več nadzora, kot se zavedamo. Premagovanje strahu in panike nam bosta dala dodatno zmogljivost v življenju, da se osredotočimo, na kar si res želimo početi in biti. Potreben je napor, a predstavljajmo si, kako bomo nagrajeni.

Ko nas grabi panika, uporabimo naslednji pristop. Sprejmimo paniko. Ne poizkusimo se upirati. Bodimo pozorni na paniko. Ko jo opazimo, ocenimo raven strahu in začnimo izdihovati dlje, kot pa vdihovati. Nadaljujmo z govorjenjem ali obnašanjem, kot da je vse normalno. S tem podzvesti pošljemo močan signal, da tako dramatičen odziv ni potreben, saj se ne dogaja nič nenavadnega. Kot gasilci, ki prispejo na kraj nesreče, ugotovijo, da ni nobene krize, in se vrnejo v gasilni dom. Če je potrebno, zgornje korake ponovimo v mislih. Pričakujmo najboljše. Eden največjih občutkov v življenju je dognanje, da lahko strah nadzorujemo veliko bolj, kot smo si sprva predstavljali.

Slika 6.6.23: V trenutku panike uporabimo pomirjevalni pristop[22].



## **Poglavje 7      Sklepne ugotovitve**

Mnenja smo, da ima aplikacija ogromen potencial, ki ga trenutno nismo zmožni doseči sami, zato je aplikacija le delček tistega, kar bi lahko bila. Razvili smo jo v prvi vrsti iz osebnih razlogov. Želeli smo ustvariti nekaj, kar bo imelo pozitiven vpliv na nas, našo družino in prijatelje. Končni produkt ni primeren le za nas, temveč za vsakega človeka, ki se zaveda, da so misli in občutki rezultat našega psihološkega stanja, ki ga s trudom in časom lahko spremenimo.

Aplikacija lahko veliko prispeva družbi. Če jo uporabljajo otroci in mladostniki, jim ni potrebno čakati več desetletij, da se naučijo istih nauk in idej, ki jih v življenju razkrijejo njihovi straši, ampak lahko nadgrajujejo in za naslednjo generacijo ustvarjajo čisto nova, sedaj še nepoznana psihološka razkritja. Tako mladi kot odrasli si lahko pomagajo v primeru krize identitete. Sploh danes je to velik problem, ko dajemo vedno večjo vrednost materialnim dobrinam, sebe pa zapostavljamo. S pomočjo aplikacije spoznajo sebe in svoja čustva ter ugotovijo, da materialnost ni pomembnejša od tega, kar nosijo v sebi. Poleg večanja vrednosti materialnih dobrin je pogosto težava tudi v preveliki odvisnosti med ljudmi. Ko se na nekoga preveč navežemo, lahko od njega postanemo odvisni, ko nas zapusti, pa imamo občutek, da smo izgubili del sebe. Tudi tu lahko aplikacija prispeva pomembna razkritja, ki človeku pomagajo, da v odnosih postavi trdne meje, ki se jih ne prestopa, če pa jih, se tega vsaj zaveda.

Najbolj osnoven namen aplikacije je uporabniku ponuditi odmerek pozitivnosti, kadarkoli ga potrebuje. Želimo, da se vrača vsak dan in da se po vsaki uporabi počuti bolje. Vsakič, ko doživi nekaj, česar ne razume, ali pa ne ve, kako se s tem spoprijeti, ima priložnost, da ne ostane ujet v svoji glavi, temveč odkrije nekaj novega, kar mu odpre svet. S časom bo razvil svojo čustveno inteligenco in s svojimi idejami bo začel vračati okolju, ki mu je ponudilo priložnost osebnostne rasti. Aplikacijo lahko uporabljajo otroci in mladostniki, ki sveta še ne poznajo in si niti ne predstavljajo razsežnosti, ki jih ponuja njihov um. Prav tako jo lahko uporabljajo odrasli ljudje, ki imajo stresne službe in odnose, travme iz preteklosti ali karkoli drugega, kar jim otežuje življenje v smislu psihe. Nenazadnje lahko aplikacijo uporabljajo tudi ljudje, ki pomoči ne potrebujejo, so pa dovolj pametni, da se zavedajo pomena psihične stabilnosti in so morda sami pripravljeni pomagati ljudem v stiski.

Diplomsko delo povezuje računalništvo in psihologijo. Internet je v našem primeru medij za prenos idej, ki doseže veliko več ljudi, kot to lahko storijo fizični mediji, na primer knjige. To nam seveda pomaga, da se dotaknemo ogromno ljudi, a zraven prinaša druge težave. Kot smo omenili v eni od osnovnih rezin, se lahko ideje nenadzorovano širijo kot plamen, še toliko bolj danes, ko smo le klik stran od informacij. Ljudje imamo nevarno nagnjenje, da verjamemo skoraj vsemu, kar preberemo, če pa se teorija v praksi ne izide, pogosto dvomimo v svoje sposobnosti in ne v prebrane ideje. Ponujanje psihološke pomoči je pravzaprav poseg v intimno stran človeka, ki jo skrivamo. Veliko ljudi, ki bodo aplikacijo uporabljali, se bo za to odločilo, ker se morda počutijo šibke in ranljive ter si želijo najti vsebino, ki bi jim pomagala. To je seveda namen aplikacije, vendar se s psihologijo ranljivega uporabnika ni za igrati, sploh če je dovzeten za negativne ideje. Vsak uporabnik oziroma čutilec lahko piše svoje ideje, ki pa morda odražajo njegovo negativno plat. Če taka ideja pride pred oči ranljivega čutilca, lahko povzroči ogromno škodo, zato bi bilo najbolje, da vsako idejo pred objavo pregleda strokovnjak psihologije, ki je sposoben oceniti, kakšen vpliv ima ta lahko na posameznika. Razlog več za to je tudi ta, da psihologi v praksi ne uporabljajo takega pristopa, temveč obravnavano osebo najprej preučijo z uporabo različnih psiholoških testov, ki pokažejo njegovo psihično stanje. Šele po tem se odločijo, kaj je najboljši način, da se obravnavani osebi pomaga. V aplikaciji smo se temu izognili tako, da ponujamo ideje za različne vrste mišljenja, vendar to pomeni, da mora uporabnik sam vedeti, kaj potrebuje, ali pa to odkriva sproti. Sistem ima zato očitne pomanjkljivosti, na katere bomo morali biti pozorni pri nadaljnjem razvoju.

Projekta razvoja aplikacije smo se lotili brez konkretnega predhodnega znanja. Lahko bi izbrali tehnologije, ki so že dolgo časa v uporabi, a če smo se bili primorani učiti od začetka, naj bodo to tehnologije, ki definirajo nove generacije. Node.js in Express.js smo uporabili le kot vmesnik med podatkovno bazo in odjemalcem, zato imamo namen v prihodnosti uporabiti njuno polno moč. Možnost, da v celotni aplikaciji uporabljamo le en programski jezik, se nam zdi izredno plodna. Želimo si preizkusiti tudi tehnologije, kot sta Ruby on Rails ali pa Django, čeprav nista več tako popularni, sta nam pa oba jezika, tako Ruby kot Python, ki ju ti tehnologiji uporabljata, izredno privlačna.

Pri podatkovni bazi smo uporabili ogromno denormalizacije, kar pomeni, da dokumentov v zbirkah nismo referencirali s ključem, kot se to počne pri mehanizmu SQL, temveč smo informacije dokumentov vedno znova ponavljali. Primer je rezina, ki jo doda vsak čutilec. Dokument te rezine je sedaj v zbirki dodanih rezin vseh čutilcev, kar se sliši zelo redundantno, a protiutež je ta, da nam ni potrebno vsake rezine posebej potegniti iz podatkovne baze. Rezultat tega naj bi bilo zelo hitro poizvedovanje, vendar obstaja možnost, da v našem primeru ni tako, saj je shema podatkovne baze, ki smo jo zasnovali, daleč od

popolne, in bi morda poizvedovanje delovalo še hitreje, če bi uporabili mehanizem SQL, s katerim smo imeli že izkušnje. Ideja, da sheme ni potrebno definirati, se lahko začetniku zdi dobra, vendar se slej kot prej prikažejo posledice premalo podrobnega načrtovanja. V končani shemi podatkovne baze je opaziti pomanjkanje vidika varnosti, poleg nastavljanja edinstvenih vrednosti in njihovih tipov. Varnosti v mehanizmu NoSQL nismo dovolj raziskali in je absolutno eden glavnih področij, ki se jim moramo še posvetiti.

Najbolj zadovoljni smo z uspehom uporabe ogrodja AngularJS. Učenju delovanja le-tega smo posvetili res ogromno časa in večkrat se je pojavil dvom o njegovi uporabi. Obstaja več veliko bolj enostavnih načinov izdelave odjemalnega dela, v skrajnem primeru bi lahko uporabili sintakso HTML brez posebne nadgradnje, vendar smo si zares želeli stopiti v korak s časom. AngularJS nam je omogočil zelo hitro realizacijo zamišljenih funkcionalnosti, izkazal pa se je za odlično ogrodje pri razvoju visoko interaktivne aplikacije. Kljub vsemu je treba poudariti, da bi lahko izbrali še boljšo pot. Obstajajo tehnologije, ki se razvijanja odjemalnega dela lotevajo na drugačen način, kar nas sicer ne bi zanimalo, če ne bi bila v razvoju tudi naslednja različica ogrodja AngularJS, ki uporablja prav te novejšje principe. Podpora za prvo različico, kot kaže, že počasi upada, kar pomeni, da bomo morali slej kot prej nadgraditi znanje, v najslabšem primeru pa je bilo učenje prve različice AngularJS zaman. Vseeno bomo obdržali občutek, da smo se zmožni naučiti katerega koli ogrodja, ter ga uspešno uporabiti v praksi.

## **7.1 Odzivi testnih uporabnikov**

Preden smo aplikacijo testirali z uporabniki, smo nekaj ljudi povprašali o ustreznosti naših osnovnih rezin. Glede na to, da smo relativno mladi in nismo študirali psihologije, je bila večina nad vsebino pozitivno presenečena. Čudili so se, kako da razumemo stvari, ki so jih oni spoznali šele pri petdesetih letih, a to je seveda posledica tega, da se danes zelo malo poudarka daje na psihološko stanje ljudi. K sreči je odnos do tega iz leta v leto boljši.

Aplikacijo smo izpostavili na socialnih omrežjih in forumih. Obiskalo jo je kar nekaj ljudi, a smo dobili občutek, da je bila večina na splošno malo zadržana. Čustva in občutki so nekaj zelo osebnega in ni se vsakemu lahko soočiti z njimi, zato mora biti zelo očitno, da je namen aplikacije izključno pozitiven in da je uporabnik med uporabo popolnoma varen. Ocenjujemo, da se je le petina registriranih uporabnikov vrnila po prvem obisku, kar je za nas razumljivo. Vsi so spremenili misel in dodali vsaj eno rezino, a nekaterih osebnostna rast in razumevanje čustev enostavno ne zanima, nekateri na to niso pripravljeni, a se bodo nekoč le vrnili, obstajajo pa taki, ki so v konceptu aplikacije takoj videli vrednost in so tudi napisali kakšno svojo rezino. Taki so nam povedali tudi, da se jim aplikacija zdi izredno motivacijska.

Ves čas smo imeli pogled na podatkovno bazo, tako da smo videli, kaj počnejo uporabniki. To se nam sicer sliši kot nespodobno dejanje, a ker smo vpisovanje osebnih informacij primerno omejili, nismo nikoli vedeli, kdo so uporabniki v resnici. Skrb glede tega so nam izrazili nekateri uporabniki, ki so mnenja, da je na internetu že tako slabo poskrbljeno za varnost podatkov, sedaj pa želimo iz njih pridobiti še najbolj osebne podatke. Strinjali smo se, da je skrb primerna in v prihodnosti je potrebno glede tega dobro premisliti. Literatura[20] sicer pravi, da je premalo ljudi pripravljenih pokazati svoja čustva, kar lahko slabo vpliva na njih in ljudi okoli njih. Iz podatkov smo videli, da so si uporabniki takoj spremenili misel, s čimer smo bili izredno zadovoljni. Za nas to pomeni, da uporabnik takoj vidi, da je stvar interaktivna in da si jo lahko priredi po svojih željah.

Dobili smo veliko odzivov na barvo in obliko aplikacije, ki se je vsem zdela dobra in primerna. Dobili smo tudi odzive na funkcionalnost aplikacije in veliko mnenj smo takoj upoštevali. Širina pogledov je bila preširoka in med elementi je bilo preveč prostora, zato smo širino zmanjšali. Uporabniki so bili zbegani nad praznim prikazom zadnjih aktivnosti, zato se ta element sedaj pokaže šele takrat, ko kakšne aktivnosti dejansko obstajajo. Koncept dosežkov jim ni bil dovolj razumljiv, zato bo potrebno premisliti, kako za njih ustvariti večji pomen in čemu dejansko služijo slike v profilki. Sama struktura rezin jim je bila razumljiva in so si jih tudi veliko dodali, nekateri pa so celo ustvarili zelo dobre rezine, ki so takoj vplivale tudi na naše mišljenje. To je dokaz, da stvar deluje, in prepričani smo, da imajo naše osnovne rezine enak vpliv na uporabnike, kot imajo njihove rezine na nas.

## **7.2 Cilji za nadaljnji razvoj**

Iz mnenj uporabnikov smo razvili dobro podlago za nadaljnje delo. Aplikacijo bi bilo dobro razviti še v angleščini in jo tako ponuditi tudi izven okvirjev Slovenije. Dobro bi bilo, da se razvije nekakšna klepetalnica ali forum, da se ima uporabnik možnost z nekom pogovarjati. Tako se ne bi vračal le zaradi vsebine, temveč tudi zaradi prijateljev. Na voljo bi lahko bila spletna psihološka svetovalnica, kjer strokovnjak odgovarja na vprašanja uporabnikov, ki imajo specifične težave. Mogoče bi celo dopustili možnost, da na vprašanja odgovarjajo drugi uporabniki, ki imajo ogromno dosežkov in so se v skupnosti znotraj aplikacije izredno izkazali.

Z aplikacijo smo zadovoljni, ker se zavedamo, koliko časa smo vanjo vložili, odzivi so bili pozitivni, pot pa je še dolga, saj ima ogromna neizkoriščenega potenciala. Namesto podatkovne baze MongoDB si želimo v aplikaciji preizkusiti MySQL. V preteklosti smo imeli z njo veliko uspeha in zanima nas, kako bi se obnesla pri takem tipu aplikacije, kjer se

hitro izmenjuje veliko podatkov. MongoDB nam je tu dobro služila, zato želimo nekoč nadgraditi tudi znanje podatkovnih baz, ki uporabljajo mehanizem NoSQL.

Obstaja veliko tehnologij, ki omogočajo sprotno testiranje razvite kode, a se nam je ta koncept zdel prevelik zalogaj, ki bi le upočasnil napredek. To razmišljanje verjetno ni bilo tako napačno, a danes zares vidimo vrednost sprotnega testiranja. Velikokrat je aplikacija brez razloga nehala delovati ali pa implementacija nove funkcionalnosti ni bila uspešna. Porabili smo tudi po več ur naenkrat, da smo na koncu odkrili, da v eni vrstici kode pri spremenljivki manjka črka. Testiranje je eden prvih konceptov, ki jih bomo v nadaljevanju implementirali.

Z vidika razvoja nas najbolj moti dejstvo, da velikost aplikacije v brskalniku ni fleksibilna na različnih napravah. Narejena je za računalniški ekran in čeprav brskalniki danes že sami ponujajo malo fleksibilnosti, to ni isto, kot če bi bila aplikacija zasnovana za vse vrste ekranov. Vsekakor je tudi to koncept, ki ga želimo čim prej implementirati.

Na strani odjemalca imamo namen preizkusiti nova ogrodja, ki uporabljajo nove principe in pridobivajo vedno večjo podporo. Naučili smo se uporabljati ogrodje AngularJS, sedaj pa mu že grozi zastarelost. Tega ne obžalujemo, nasprotno, navdihuje nas ideja, da se svet računalništva burno razvija, in z veseljem se bomo naučili uporabljati ogrodje AngularJS 2 ali katerega drugega.

V korist uporabnikov si želimo, da bi vse rezine pregledal usposobljeni strokovnjak psihologije. Tako bi bili prepričani, da te uporabnikom ne škodijo. En način, ki ga lahko hitro uvedemo, je ta, da bi morali vsako ustvarjeno rezino najprej potrditi mi, preden bi ta bila vidna drugim uporabnikom, pozneje pa bi dobili nekoga, ki je za tako odločanje dejansko kvalificiran.

Zaradi ogromne količine dela aplikacija ne vsebuje zelo pomembne funkcionalnosti odstranjevanja in spreminjanja rezin. Brez tega v praksi ne gre, čeprav se nad tem zaenkrat ni pritoževal noben od uporabnikov. Imeli smo ogromne težave pri razvijanju, zato smo se odločili, da to funkcionalnost izpustimo. Do težav je prišlo, ker nismo sprotno testirali kode in nismo točno razumeli, kakšne so pravice in zahteve pri modulih za avtentikacijo uporabnika, tako da je to še eden od konceptov, ki jih moramo bolj podrobno raziskati.

## Literatura

- [1] Ljubica Uvodić-Vranić, *Pustolovščina osebne preobrazbe*, Ebesede, str. 11-28, 34-47, 82-86, 118-137 in 157-160, 2014.
- [2] Node.js. [Online]. Dosegljivo: <https://nodejs.org/en/>.
- [3] Chrome V8. [Online]. Dosegljivo: <https://developers.google.com/v8/>.
- [4] Uvod v Node.js. [Online]. Dosegljivo: [http://www.tutorialspoint.com/nodejs/nodejs\\_introduction.htm](http://www.tutorialspoint.com/nodejs/nodejs_introduction.htm).
- [5] Upravljalac paketov Node. [Online]. Dosegljivo: <https://www.npmjs.com/>.
- [6] Express.js. [Online]. Dosegljivo: <http://expressjs.com/>.
- [7] MongoDB. [Online]. Dosegljivo: <https://www.mongodb.org/>.
- [8] Definicija MongoDB. [Online]. Dosegljivo: <http://searchdatamanagement.techtarget.com/definition/MongoDB>.
- [9] Mongoose.js. [Online]. Dosegljivo: <http://mongoosejs.com/>.
- [10] Passport. [Online]. Dosegljivo: <http://passportjs.org/>.
- [11] JavaScript Web Token. [Online]. Dosegljivo: <https://jwt.io/>.
- [12] AngularJS. [Online]. Dosegljivo: <https://docs.angularjs.org/guide/introduction>.
- [13] JavaScript. [Online]. Dosegljivo: <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Introduction>.
- [14] jQuery. [Online]. Dosegljivo: <http://www.tutorialspoint.com/jquery/jquery-overview.htm>.
- [15] HTML. [Online]. Dosegljivo: <http://www.computerhope.com/jargon/h/html.htm>.
- [16] CSS. [Online]. Dosegljivo: <http://html.net/tutorials/css/lesson1.php>.

- [17] Git. [Online]. Dosegljivo: <http://git-scm.com/video/what-is-git>.
- [18] Napoleon Hill, *Think & Grow Rich*, Ballantine Books, str. 1-14 in 177-184, 1983.
- [19] David Deida, *The Way of the Superior Man*, Sounds True, str. 14-50, 2004.
- [20] David Wygant, *Naked!*, Hay House, str. 15-36, 2012.
- [21] Understanding & Overcoming Fear. [Online]. Dosegljivo: <http://www.eruptingmind.com/understanding-overcoming-fear/>.
- [22] 5 Sure-fire Ways to Overcome Fear and Anxiety Today. [Online]. Dosegljivo: <http://www.uncommonhelp.me/articles/overcome-fear-and-anxiety/>





